

Interfaces for Modular Surgical Planning and Assistance Systems

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

DISSERTATION

zur Erlangung des Akademischen Grades

DOKTOR-Ingenieur
(Dr.-Ing.)

im Fachgebiet

INFORMATIK

vorgelegt von

Dipl.-Inform. Michael Gessat

geb. am 6. April 1979 in Bad Friedrichshall

Die Annahme der Dissertation wurde empfohlen von:

1. Prof. Dr.-Ing. Gerik Scheuermann, Institut für Informatik, Universität Leipzig
2. Prof. Dr.-Ing. Heinz Wörn, Fakultät für Informatik, Universität Karlsruhe (TH)

Die Verleihung des akademischen Grades erfolgt mit Bestehen
der Verteidigung am 22.06.2010 mit dem Gesamtprädikat "magna cum laude".

"Those about the patient must present the part to be operated upon as may seem proper, steady, in silence, and listening to the commands of the operator."

Hippocrates of Kos (460 – 377 BC), *On The Surgery*

Abstract

Modern surgery of the 21st century relies in many aspects on computers or, in a wider sense, digital data processing. Department administration, OR scheduling, billing, and – with increasing pervasion – patient data management are performed with the aid of so called Surgical Information Systems (SIS) or, more general, Hospital Information Systems (HIS).

Computer Assisted Surgery (CAS) summarizes techniques which assist a surgeon in the preparation and conduction of surgical interventions. Today still predominantly based on radiology images, these techniques include the preoperative determination of an optimal surgical strategy and intraoperative systems which aim at increasing the accuracy of surgical manipulations.

CAS is a relatively young field of computer science. One of the unsolved "teething troubles" of CAS is the absence of technical standards for the interconnectivity of CAS system. Current CAS systems are usually "islands of information" with no connection to other devices within the operating room or hospital-wide information systems. Several workshop reports and individual publications point out that this situation leads to ergonomic, logistic, and economic limitations in hospital work. Perioperative processes are prolonged by the manual installation and configuration of an increasing amount of technical devices. Intraoperatively, a large amount of the surgeons' attention is absorbed by the requirement to monitor and operate systems. The need for open infrastructures which enable the integration of CAS devices from different vendors in order to exchange information as well as commands among these devices through a network has been identified by numerous experts with backgrounds in medicine as well as engineering.

This thesis contains two approaches to the integration of CAS systems:

- For perioperative data exchange, the specification of new data structures as an amendment to the existing DICOM standard for radiology image management is presented. The extension of DICOM towards surgical application allows for the seamless integration of surgical planning and reporting systems into DICOM-based Picture Archiving and Communication Systems (PACS) as they are installed in most hospitals for the exchange and long-term archival of patient images and image-related patient data.
- For the integration of intraoperatively used CAS devices, such as, e.g., navigation systems, video image sources, or biosensors, the concept of a surgical middleware is presented. A c++ class library, the TiCoLi, is presented which facilitates the configuration of ad-hoc networks among the modules of a distributed CAS system as well as the exchange of data streams, singular data objects, and commands between these modules. The TiCoLi is the first software library for a surgical field of application to implement all of these services.

To demonstrate the suitability of the presented specifications and their implementa-

tion, two modular CAS applications are presented which utilize the proposed DICOM extensions for perioperative exchange of surgical planning data as well as the TiCoLi for establishing an intraoperative network of autonomous, yet not independent, CAS modules.

Kurzfassung

Die moderne Hochleistungschirurgie des 21. Jahrhunderts ist auf vielerlei Weise abhängig von Computern oder, im weiteren Sinne, der digitalen Datenverarbeitung. Administrative Abläufe, wie die Erstellung von Nutzungsplänen für die verfügbaren technischen, räumlichen und personellen Ressourcen, die Rechnungsstellung und – in zunehmendem Maße – die Verwaltung und Archivierung von Patientendaten werden mit Hilfe von digitalen Informationssystemen rationell und effizient durchgeführt. Innerhalb der Krankenhausinformationssysteme (KIS, oder englisch HIS) stehen für die speziellen Bedürfnisse der einzelnen Fachabteilungen oft spezifische Informationssysteme zur Verfügung. Chirurgieinformationssysteme (CIS, oder englisch SIS) decken hierbei vor allen Dingen die Bereiche Operationsplanung sowie Materialwirtschaft für spezifisch chirurgische Verbrauchsmaterialien ab.

Während die genannten HIS und SIS vornehmlich der Optimierung administrativer Aufgaben dienen, stehen die Systeme der Computerassistierten Chirurgie (CAS) wesentlich direkter im Dienste der eigentlichen chirurgischen Behandlungsplanung und Therapie. Die CAS verwendet Methoden der Robotik, digitalen Bild- und Signalverarbeitung, künstlichen Intelligenz, numerischen Simulation, um nur einige zu nennen, zur patientenspezifischen Behandlungsplanung und zur intraoperativen Unterstützung des OP-Teams, allen voran des Chirurgen. Vor allen Dingen Fortschritte in der räumlichen Verfolgung von Werkzeugen und Patienten ("Tracking"), die Verfügbarkeit dreidimensionaler radiologischer Aufnahmen (CT, MRT, ...) und der Einsatz verschiedener Robotersysteme haben in den vergangenen Jahrzehnten den Einzug des Computers in den Operationssaal – medienwirksam – ermöglicht. Weniger prominent, jedoch keinesfalls von untergeordnetem praktischen Nutzen, sind Beispiele zur automatisierten Überwachung klinischer Messwerte, wie etwa Blutdruck oder Sauerstoffsättigung.

Im Gegensatz zu den meist hochgradig verteilten und gut miteinander verwobenen Informationssystemen für die Krankenhausadministration und Patientendatenverwaltung, sind die Systeme der CAS heutzutage meist wenig oder überhaupt nicht miteinander und mit Hintergrunddatenspeichern vernetzt. Eine Reihe wissenschaftlicher Publikationen und interdisziplinärer Workshops hat sich in den vergangenen ein bis zwei Jahrzehnten mit den Problemen des Alltagseinsatzes von CAS Systemen befasst. Mit steigender Intensität wurde hierbei auf den Mangel an infrastrukturellen Grundlagen für die Vernetzung intraoperativ eingesetzter CAS Systeme miteinander und mit den perioperativ eingesetzten Planungs-, Dokumentations- und Archivierungssystemen hingewiesen. Die sich daraus ergebenden negativen Einflüsse auf die Effizienz perioperativer Abläufe – jedes Gerät muss manuell in Betrieb genommen und mit den spezifischen Daten des nächsten Patienten gefüttert werden – sowie die zunehmende Aufmerksamkeit, welche der Operateur und sein Team auf die Überwachung und dem Betrieb der einzelnen Geräte verwenden muss, werden als eine der "Kinderkrankheiten" dieser relativ jungen Technologie betrachtet und stehen

einer Verbreitung über die Grenzen einer engagierten technophilen Nutzergruppe hinaus im Wege.

Die vorliegende Arbeit zeigt zwei parallel von einander (jedoch, im Sinne der Schnittstellenkompatibilität, nicht gänzlich unabhängig voneinander) zu betreibende Ansätze zur Integration von CAS Systemen.

- Für den perioperativen Datenaustausch wird die Spezifikation zusätzlicher Datenstrukturen zum Transfer chirurgischer Planungsdaten im Rahmen des in radiologischen Bildverarbeitungssystemen weit verbreiteten DICOM Standards vorgeschlagen und an zwei Beispielen vorgeführt. Die Erweiterung des DICOM Standards für den perioperativen Einsatz ermöglicht hierbei die nahtlose Integration chirurgischer Planungssysteme in existierende "Picture Archiving and Communication Systems" (PACS), welche in den meisten Fällen auf dem DICOM Standard basieren oder zumindest damit kompatibel sind. Dadurch ist einerseits der Tatsache Rechnung getragen, dass die patientenspezifische OP-Planung in hohem Masse auf radiologischen Bildern basiert und andererseits sicher gestellt, dass die Planungsergebnisse entsprechend der geltenden Bestimmungen langfristig archiviert und gegen unbefugten Zugriff geschützt sind – PACS Server liefern hier bereits wohlerprobte Lösungen.
- Für die integration intraoperativer CAS Systeme, wie etwa Navigationssysteme, Videobildquellen oder Sensoren zur Überwachung der Vitalparameter, wird das Konzept einer "chirurgischen Middleware" vorgestellt. Unter dem Namen TiCoLi wurde eine c++ Klassenbibliothek entwickelt, auf deren Grundlage die Konfiguration von ad-hoc Netzwerken während der OP-Vorbereitung mittels *plug-and-play* Mechanismen erleichtert wird. Nach erfolgter Konfiguration ermöglicht die TiCoLi den Austausch kontinuierlicher Datenströme sowie einzelner Datenpakete und Kommandos zwischen den Modulen einer verteilten CAS Anwendung durch ein Ethernet-basiertes Netzwerk. Die TiCoLi ist die erste frei verfügbare Klassenbibliothek welche diese Funktionalitäten dediziert für einen Einsatz im chirurgischen Umfeld vereinigt.

Zum Nachweis der Tauglichkeit der gezeigten Spezifikationen und deren Implementierungen, werden zwei modulare CAS Anwendungen präsentiert, welche die vorgeschlagenen DICOM Erweiterungen zum perioperativen Austausch von Planungsergebnissen sowie die TiCoLi zum intraoperativen Datenaustausch von Messdaten unter echtzeitnahen Anforderungen verwenden.

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

.....
(Ort, Datum)

.....
(Unterschrift)

Acknowledgment

I deeply thank my thesis advisor, Prof. Dr. Heinz U. Lemke for invaluable assistance, advice, and supervision. I also wish to thank Dr.-Ing. Oliver Burgert for the support he was providing, for the discussions we had, and for the confidence he placed in me. I also thank Prof. Dr. Geric Scheuermann, Prof. Dr. Heinz Wörn, Prof. Dr. Hans-Peter Fährnich, and Prof. Dr. Heyer who served as reviewers and examiners.

I thank Universität Leipzig, the German Ministry of Education and Research, the European Regional Development Fund, and the State of Saxony for the opportunities and funding I was equipped with.

I also want to thank my colleagues, co-workers, and superiors at the Innovation Center Computer Assisted Surgery in Leipzig who were an important source of inspiration and critique on so many levels and occasions.

Finally, I wish to thank my family and friends for the personal and emotional support during these challenging times.

Contents

List of Abbreviations	vi
List of Figures	viii
List of Tables	xi
1 Introduction	1
1.1 Distributed Systems	3
1.2 Distributed Systems in Medical Informatics	5
1.2.1 Examples of Modular CAS Systems	6
1.2.2 Summary	11
1.3 Aim of the Thesis	11
1.4 Structure of the Thesis	12
2 State of the Art	13
2.1 A Brief History of Computer Assisted Surgery	13
2.1.1 Image Guided Surgery	13
2.1.2 Preoperative Planning	18
2.1.3 A New Paradigm: Model Guided Therapy	20
2.2 Surgical Informatics	26
2.2.1 Integrated OR solutions and projects	29
2.2.2 TIMMS	31
2.2.3 Standards and Protocols	33
3 Dataflow in CAS	39
3.1 From Workflows to Dataflows	39
3.2 Data Exchange Requirements	42

4	Surgical DICOM	47
4.1	Identification of DICOM Work Items	48
4.2	Surface segmentation SOP Class	50
4.2.1	Requirements	52
4.2.2	Surface Segmentation Class Diagram	53
4.2.3	Surface Segmentation Storage SOP Class	54
4.3	Implant Template SOP Classes	56
4.3.1	Overview	56
4.3.2	Generic Implant Template Storage SOP Class	59
4.3.3	Implant Assembly Templates	65
4.3.4	Implant Template Groups	68
4.3.5	Implantation Plan SR Document	69
4.4	Summary	71
5	An Open-Source Interface for OR Integration	73
5.1	TiCoLi - An Overview	74
5.2	TiCoLi: Basic Types	75
5.3	The API, the Core, and the Managers	77
5.3.1	The Device Manager	77
5.3.2	The Message Manager	88
5.3.3	The Attribute Manager	95
5.3.4	The Method Manager	100
5.3.5	The Streaming Manager	105
5.4	Performance Tests	111
5.4.1	Streaming Throughput and Reliability	112
5.4.2	Messaging Speed	116
5.5	Summary and Discussion	118
6	Clinical Applications	121
6.1	Transapical Aortic Valve Implantation	121
6.1.1	Computer Assisted Transapical Aortic Valve Implantation	126
6.1.2	Infrastructure	126
6.2	Cortical Stimulation and Mapping	132
6.2.1	The Central Sulcus	132
6.2.2	Localization of the Central Sulcus	133

6.2.3	Intraoperative Mapping of the Central Sulcus	135
6.2.4	Preoperative Model Generation	138
6.3	Summary and Discussion	141
7	Conclusion & Outlook	143
7.1	Summary	143
7.2	Conclusion	145
7.3	Outlook	146
A	Data Flow Diagrams	I
A.1	Diagram Types	II
B	DICOM	V
B.1	The DICOM Information Model	V
B.2	DICOM Information Objects	V
B.3	DICOM Messages	IX
B.4	DICOM Services	XII
C	TiCoLi Protocols and Libraries	XIII
D	Algorithms and Implementation Details	XVII
D.1	HandleSets	XVII
D.2	Thread Safe Callbacks	XVIII
D.3	The Gesture Detection Module	XXII
E	S-DICOM IODs	XXXI
E.1	The Surface Segmentation IOD	XXXI
E.2	The Implant Template IOD	XXXVIII
E.3	The Implant Assembly Template IOD	XLVIII
E.4	The Implant Template Group IOD	LI
	Bibliography	LIII

List of abbreviations

ADL	Archetype Definition Language
AE	Application Entity (DICOM terminology)
API	Application Programming Interface
AV	Aortic Valve
AVR	Aortic Valve Replacement
cDFD	current Data Flow Diagram
CT	Computed (X-ray) Tomography
CAS	Computer Assisted Surgery
CSF	Cerebrospinal Fluid
DICOM	Digital Imaging and Communications in Medicine
DTI	Diffusion Tensor Imaging
DFD	Data Flow Diagram
DNS	Domain Name System
DNS-SD	DNS Service Discovery
EBS	Electronic Billing Systems
EHR	Electronic Health Records
FOPL	First Order Predicate Logic
FOR	Frame of Reference (DICOM terminology)
GEHR	Good Electronic Health Records, formerly Good European Health Records
GOL	General Ontological Language
gPM	Generic Patient Model
GUID	Globally Unique Identifier (UID in DICOM terminology)
HIS	Hospital Information System(s)
HL7	Health Level Seven
ICCAS	Innovation Center Computer Assisted Surgery

IE	Information Entity (DICOM terminology)
IGS	Image Guided Surgery
IHE	Integrating the Healthcare Enterprise
IOD	Information Object Definition (DICOM terminology)
IP	Internet Protocol
LIMS	Laboratory Information Management System(s)
MAF	Multimod Application Framework
MVR	Mitral Valve Replacement
mDNS	Multicast DNS
MGS	Model Guided Surgery
MR, MRI	Magnetic Resonance (Imaging)
NNS	Nearest-Neighbor-Search
OR	Operating Room
PACS	Picture Archiving and Communication System
pDFD	proposed Data Flow Diagram
PM	Patient Model
PSM	Patient Specific Model
RIM	Reference Information Model
RP	Rapid Pacing
RTP	Real-Time Transport Protocol
SSEP	Somatosensory Evoked Potential
SM	Situation Model
SOP	Service-Object-Pair (DICOM terminology)
S-PACS	Surgical PACS
S-WF	Surgical Workflow
PSM	PatientSpecific Model
TA-AVI	Transapical Aortic Valve Implantation
TCP	Transmission Control Protocol
TiCoLi	TIMMS Communication Library
TIMMS	Therapy Imaging and Model Management System
TROT	Target Region of Treatment
UDP	User Datagram Protocol
UID	Unique Identifier (DICOM Terminology)
US	Ultrasound

List of Figures

1.1	NC FESS Architecture	7
1.2	LOCALITE BrainNavigator Architecture	8
1.3	da Vinci [®] Surgical System	10
1.4	Augmented Reality in the da Vinci [®] Console	10
2.1	Structure of TIMMS	32
2.2	Surgical Workflow for Mitral Valve Reconstruction	35
3.1	Workflow and Dataflow for MVR	40
3.2	Current Dataflow in CAS	41
4.1	DICOM Model of the Real World Including Surfaces	51
4.2	UML Class Diagram of the Surface Segmentation	53
4.3	Surface Segmentation IOD Instance	55
4.4	Generic Workflow for Implantation Planning	57
4.5	DICOM Information Model for Implantation Planning	58
4.6	UML Class Diagram of the Generic Implant Template and its Modules	59
4.7	UML class Diagram of the Generic Implant Template	60
4.8	Relations Between Original, Replaced, and Derived Implant Templates	61
4.9	UML Class Diagram of the Generic Implant Template 2D Drawings and 3D Models Modules	62
4.10	Planning Landmarks in Stented Aortic Valve Implant Templates	63
4.11	UML Class Diagram of the Generic Implant Template Planning Landmarks Module	64
4.12	UML Class Diagram of the Generic Implant Template Mating Features Module	65
4.13	UML Class Diagram of The Implant Assembly Template Module	67
4.14	Implant Templates, Mating Features, and an Implant Assembly Template	69

4.15	Implant Template Group	70
4.16	UML Class Diagram of the Implant Template Group Module	71
4.17	Structure of a DICOM Implant Plan SR Document	71
5.1	TiCoLi Protocol Stack	75
5.2	UML Class Diagram of the TiCoLi API, Core, and Managers	78
5.3	UML Class Diagram of the TiCoLi DeviceManager and Related Classes	79
5.4	UML Sequence Diagram of Device Discovery	83
5.5	Device Handles	84
5.6	UML Sequence Diagram of Service Discovery	86
5.7	Service Handles	87
5.8	UML Class Diagram of the TiCoLi MessageManager and Related Classes	89
5.9	UML Sequence Diagram of Session Initialization	93
5.10	Message Distribution	95
5.11	UML Class Diagram of the TiCoLi AttributeManager and Related Classes	96
5.12	UML Sequence Diagram of Attribute Access	99
5.13	UML Class Diagram of the TiCoLi Method Manager and Related Classes	101
5.14	UML Sequence Diagram of a Remote Method Call	104
5.15	UML Class Diagram of the TiCoLi Streaming Manager and Related Classes	105
5.16	Relation Between Instreams and Outstreams	107
5.17	UML Class Diagram of the TiCoLi Frame and Related Classes	108
5.18	UML Sequence Diagram of a TiCoLi Stream	110
5.19	Network Topologies for Framerate Measurements	112
5.20	Measured Framerates for TiCoLi Streaming	114
5.21	Measured Framerates for TiCoLi Streaming Through Bottleneck	114
5.22	Frame Drop Ratio of TiCoLi Streams	115
5.23	Network Topologies for Message Speed Tests	117
5.24	Measured Message Transfer Times	117
6.1	Catheter Aortic Valve Implants	122
6.2	Top Level Workflow of TA-AVI	123
6.3	Transapical Implantation of Sapien Valve	124
6.4	Transapical Implantation of Embracer Valve	125

6.5	TA-AVI Implantation Planning Results	127
6.6	Tracking of Coronary Ostia and Stent	127
6.7	pDFD of TA-AVI	128
6.8	Digital Aortic Valve Implant Templates	131
6.9	Principal Regions of the Cortex	134
6.10	Intraoperative View Onto the Cortex	134
6.11	Phase Reversal	135
6.12	Intraoperative System Setup for Sulcus Centralis Mapping	136
6.13	Localization of the Grid Electrode and the Central Sulcus	136
6.14	<i>SulcusMapper</i> Vizualisation	137
6.15	Collaboration Diagram for Preoperative Cortex Model Generation	139
6.16	Preoperative Cortex Extraction: Sequence Diagram	140
A.1	Gane and Sarson's DFD Notation	III
B.1	Major Structures of DICOM Information Model	VI
B.2	Structure of Information Object Definitions	VII
B.3	DICOM Model of the Real World	VIII
B.4	DICOM Information Model	IX
B.5	DIMSE Operation and Notification Flow	X
B.6	DIMSE Service Primitives	X
C.1	Structure of an OpenIGTLink Message	XIII
D.1	UML Sequence Diagram for TiCoLi Callback Execution	XIX
D.2	Architecture of the Gesture Detection Module	XXIII
D.3	Point Gesture Detection State Machine	XXVI
D.4	Bucket Search	XXVII
D.5	Bucket Search for Loops	XXIX

List of Tables

2.1	Surgical Workflows in WG24 White Paper	36
3.1	Data Types for THR, MVR, TA-AVI, NC-FESS, and BTS	45
4.1	DICOM SOP classes applicable for THR, MVR, T-AVI, NC-FESS. and BTS	48
6.1	DICOM SOP classes for data exchange in computer assisted TA-AVI .	129
B.1	DIMSE Services	XII
C.1	OpenIGTLink Header Format	XIV
E.1	Module Table for the Surface Segmentation IOD	XXXI
E.2	Surface Segmentation Module Attributes	XXXIII
E.3	Surface Mesh Module Attributes	XXXV
E.4	Algorithm Code Macro	XXXV
E.5	Points Macro	XXXVI
E.6	Vectors Macro	XXXVII
E.7	Surface Mesh Primitives Macro	XXXVII
E.8	Module Table for the Generic Implant Template IOD	XXXVIII
E.9	Generic Implant Template Description Module Attributes	XL
E.10	Generic Implant Template Derivation and Versioning Module Attributes	XLI
E.11	Generic Implant Template 2D Drawings Module Attributes	XLII
E.12	Generic Implant Template 3D Models Module Attributes	XLII
E.13	Generic Implant Template Mating Features Module Attributes	XLIV
E.14	Generic Implant Template Planning Landmarks Module Attributes . .	XLVI
E.15	Planning Landmark Point Macro	XLVI
E.16	Planning Landmark Line Macro	XLVII

E.17 Planning Landmark Plane Macro	XLVII
E.18 Module Table for the Implant Assembly Template IOD	XLVIII
E.19 Implant Assembly Template Module Attributes	L
E.20 Module Table for the Implant Template Group IOD	LI
E.21 Implant Template Group Module Attributes	LII

Chapter 1

Introduction

Modern surgery relies on computers in many aspects ranging from department and patient data management to preoperative planning and intraoperative assistance. Computer Assisted Surgery (CAS) is a name given to a set of techniques which include the usage of computers in the preparation or the actual conduction of surgical interventions. Several key technologies which were invented during the 20th century paved the way for the development of CAS techniques. Besides, obviously, the invention of the computer these technologies include tomographic and volumetric imaging as well as advances in digital image processing and computer vision, optical and magnetic tracking, robotics, and physics simulation (especially mechanical simulation) based on the Finite Element Method (FEM). The rapid progress in the development of new surgical techniques and disciplines, especially in minimally invasive surgery, produced an increasing demand for technical support of surgical perception and manipulation: with the invention of endoscopic surgery, interventional radiology, catheter interventions, and other minimally invasive techniques, the horizon of what is possible with surgery was widened and at the same time the incisions through which surgeons perceive and operate on the internal organs were constantly decreased [Jolesz, 1997]. These limitations were one of the major motivations for the development of CAS systems and techniques with the aim to augment the perceptive and dexterous capabilities of the human operator. Driven by these factors, computer-based systems were developed for manifold applications, including preoperative modeling and simulation systems, intraoperative imaging, tracking, and guidance systems, robotic tools, and tool holders.

CAS is a relatively young field of computer science. During the last 15 to 30 years, rapid progress was made in the development of CAS systems, mostly driven by research projects and small businesses aiming at very specific applications or market niches. In the terminology of Rogers' bell curve [Rogers, 2003], a few pioneer CAS technologies such as surgical navigation, image based computer assisted intervention planning, and intraoperative imaging are in the *leading edge* phase of innovation diffusion where *early majority* customers begin to adopt a new technology. Other techniques, such as surgery simulation, surgical robotics, and multimodal patient modeling, are still in the *bleeding edge* phase which is characterized by experimental systems used by *innovators* and *early adopters* who are willing to take risks and to bring up

the required personal effort that goes along with applying a not 100% mature system in the field. Before a technology becomes adopted by the majority of a population, the "teething troubles", which the technophile early adopters are willing to accept, need to be solved.

One of the unsolved "teething troubles" of CAS is the absence of technical standards for the interconnectivity of CAS systems. During the highly innovative phase of CAS technology and techniques, only little attention was spent on the development of technical infrastructures for the integration of CAS systems into the Operating Room (OR) and for data exchange between the OR and external information systems. Instead, the focus of attention lay on pushing the bar of what was technically possible to achieve with rapidly developed prototype systems and specialized products. Research prototypes as well as the commercially available CAS systems are at present usually "stand-alone island of information" [Cleary & Kinsella, 2004] with no connection to other devices within the OR, let alone hospital-wide information systems. Recent interdisciplinary workshops with clinical as well as technical experts from all major industrial countries [Jolesz & Shtern, 1992; Di Gioia *et al.*, 1996; Cleary, 1999; Haller *et al.*, 2002; Cleary & Kinsella, 2004] as well as several publications in high-ranking journals, e.g. [Taylor *et al.*, 1996; Jolesz, 1997; Jolesz *et al.*, 2001; Deinhardt, 2003; Patkin, 2003; Lemke & Vannier, 2006] made clear that the integration of technical systems inside the OR into a local area network and, on a higher level, the integration of this network with hospital-wide information systems is one of the keys to a better pervasion of CAS.

The consensus of these publications is that the increasing pervasion of CAS technology gives rise to ergonomic, economic, and logistic issues:

- Setup times in the OR are rising due to the increasing number of devices which have to be set up and configured.
- Fluent OR processes are impaired by the limitations regarding access from inside the OR to patient information which is residing in external repositories.
- The workload of surgeons, assistants, nurses, and OR technicians that comes from the requirement to monitor and maintain technical devices has continuously increased during the past decades. The lack of standardized and centralized user interfaces amplifies this effect.
- The workload of the surgeons that comes from the requirement to conceive and integrate data from various sensors and imaging modalities is continuously increasing with the accelerating progress in sensor and imaging technology and their introduction to the OR.
- The incompatibility of CAS systems forces health care provider to invest in multiple systems with similar functionalities for different use cases.

The authors of the cited reports and papers agree in the conviction that with mono-

lithic and highly specialized stand-alone systems these limitations cannot adequately be overcome.

Most of the established CAS methods depend on patient images as *the* predominant source of patient specific information for treatment planning, decision support, and intraoperative guidance. In recent years, researchers have begun to complement the image-centric view of Image Guided Surgery (IGS) by integration of non-image information and multimodal images into a comprehensive Patient Model (PM) and a Situation Model (SM). Model Guided Therapy (MGT) [Niederlag *et al.*, 2008] depends on the ability to physically bring information from various sources together and to logically integrate it with into a comprehensive model by fusing the data with pre-operative data in a modeling system. This development intensifies the importance of developing modular and flexible system architectures and open standards for system interoperability.

The recommendations for future development in CAS given by the cited workshops and publications can be summarized as follows:

- Define standardized file formats and transfer services for data exchange among devices inside the OR and external repositories.
- Develop a plug-and-play mechanism including auto-configuration of distributed CAS systems based on capability descriptions which are exchanged among the components.
- Develop an infrastructure for the integration of sensor and image data into a coherent representation of the patient.

These recommendations aim at the evolution from monolithic systems for specific applications to distributed systems created by combining the functionalities of networked modules. In the following section, the concept of distributed systems will be introduced from a computer science standpoint.

1.1 Distributed Systems

Modular design is an engineering principle where the functionalities of a system are separated into distinct entities, called modules. The separation is performed in a way which generates modules with a minimal overlap in functionality ("Separation of Concerns" [Dijkstra, 1982]). Modules are encapsulated entities, i.e. their internal logic is hidden behind a interface definition. During the process of modularization, the interfaces of each module are specified. Modules are designed to realize specific functionalities. A modular system combines the functionalities of its modules according to the requirements of a specific use case.

Modularization is applied in software as well as in hardware development for the following reasons [Meyer, 2008]:

1. Modular systems are very flexible regarding the exchange of implementations of single modules or the introduction of a new module without the need to modify or exchange any of the remaining modules in the systems.
2. Modularization has the potential to break down the complexity of system development: Instead of having a large team of developers working on one huge system, modules can be designed, implemented, and tested separately by smaller teams struggling with less complex tasks. This reduces the project management effort and risk of design faults. By developing the modules in parallel by several teams, the overall development process is potentially sped up.
3. Modularization facilitates the re-usability of source code or hardware components. Once created, a module can be used in various systems or devices which require the functionality the module implements.
4. Module interface specifications are well suited for standardization. This leads to *open modular system architectures*, i.e. modular architectures where the module and interface specifications are based on open standards or other public agreements. Open modular architectures facilitate the integration of modules which were built by different manufacturers.

Modularization can be performed to internally structure one software or hardware system which contains several modules but is still developed, sold, installed, or operated as one entity. The concept can be carried further, leading to *distributed* architectures where functionalities are spread among several autonomous systems which are developed, sold, installed, and operated independently and interact via network interfaces. Modern enterprise information systems, such as Hospital Information Systems (HIS), are usually distributed systems including a multitude of devices and workstations. Tasks and data can be sent from one workstation to another and special functionalities, such as radiological image acquisition in a hospital, or data archival are performed by specialized systems. The flexibility of distributed architectures allows systems to be dynamically adapted to the shifting requirements systems have to comply with in a preceding world: components can be exchanged, enhanced, or added with no or only minimal alterations on the remaining components. From the customers' perspective, heterogeneity regarding the selection of devices from several manufacturers is often an important requirement, especially when investing into an infrastructure which is intended to be used for several decades. A distributed system which is based on an open architecture raise no limitations in this regard. One prerequisite for such an architecture is the existence of open standards which regulate the exchange of data and commands between the modules.

The pitfalls of distributed architectures and open interfaces lie in the distribution of responsibility. Especially in environments with high safety standards, such as surgery, where systems have to be thoroughly tested in order to qualify for certification, the integration of external functionalities provided by an unknown device requires absolutely unambiguous interface definitions and well-defined quality standards. Technical

standards and open infrastructures which are the matter of this thesis are only one aspect that needs to be discussed in order to realize the idea of fully compatible digital ORs. A legal basis is required on which the distribution of responsibilities in the event of harmful failures can be founded and which regulates the process of certification of modular systems. Surgical standards need to specify the way how devices are intended to be used. Such standards have to base on evidence-based indications and counter indications for the application of certain techniques.

1.2 Distributed Systems in Medical Informatics

Modern HIS as well as the department specific HIS subsystems, such as, e.g., Radiology Information Systems (RIS) or Laboratory Information Systems (LIS) are heterogeneous distributed systems. The development of these systems which began in the 1970s was enabled by the development of a number of technical standards which specifically aim at the integration of information systems for applications in healthcare. The Health Level Seven (HL7) [HL7, Inc., 1987, 2005] standard is widely accepted among vendors of patient data management, hospital workflow management, and billing systems. It specifies a messaging protocol which can be used to transfer patient data and requests for clinical investigations or treatments, to report clinical findings, and for billing purposes. HL7 is based on a rather general information model and is primarily used to exchange information on the enterprise level. Inside the department specific subsystems of a hospital or other healthcare provider, more specific standards regulate the management of data and tasks as well as the storage of data. In Section 2.2.3, the most important of these standards are presented. Among these, the DICOM (Digital Imaging and Communications in Medicine) standard is highly relevant for CAS, especially for Image Guided Surgery (IGS). DICOM regulates the archival and exchange of radiologic images and related information in a Picture Archiving and Communication System (PACS). IGS systems which utilize radiological images for planning or guidance of surgical interventions usually operate on the basis of DICOM-encoded images. Within the PACS, images and image-related information can be exchanged between devices over a network. DICOM specifies services for the transfer of datasets and for querying a database for datasets.

In many hospitals, the accessibility to the PACS through DICOM services is restricted to the radiology department. Access to image data from other departments, such as the surgical department, is usually granted through web-interfaces which do not allow native access to the data but only to rendered jpeg images. In order to obtain a complete DICOM dataset for importing it into therapy planning software, data is usually shifted around on CDs, DVDs, or USB storages which is an inflexible, time consuming, inconvenient, and expensive solution. Some centers, after all, have started to give surgical workstations and radiotherapy planning workstations direct access to the PACS via DICOM network services [Mösges, 1993; Nuttin *et al.*, 1994; Teraea *et al.*, 1998; Law & Huang, 2003; Sectra, 2009]. On the example of digital implantation planning in orthopedic surgery, it could be shown that the efficiency of planning procedures is

fundamentally increased through direct access to the PACS database [Winter *et al.*, 2002].

1.2.1 Examples of Modular CAS Systems

In the following paragraphs, three commercially available CAS systems which are based on modular architectures are presented. Limitations in their flexibility due to missing standards in the field of CAS are discussed. In the third example, a surgical telemanipulator, two projects are presented where the modular architecture of the system was extended in research projects in order to add functionalities.

Navigated Control

The concept of Navigated Control (NC) has been introduced to Functional Endoscopic Sinus Surgery (FESS) by Lüth *et al.* [Lüth *et al.*, 2001; Koulechov *et al.*, 2005]. NC is based on commercially available optical tracking technology where tracking markers and a binocular infrared camera system are utilized to spatially register preoperative volumetric images with the situs and to localize the position of handheld pointers. Instead of a pointer, The NC FESS system tracks the position of a power tool used for cutting and suctioning in the paranasal sinuses, called *shaver*. According to the position of the tool tip in patient space, a three-planar sliced 2D representation of the CT is rendered. In addition, this position is related to a preoperatively determined working space: when the tool leaves this area, the NC control unit automatically switches off the engine that powers the tool. The working space determined by a surgeon who draws the boundaries of the safe area onto the CT images.

The NC system is based on a modular architecture (see Figure 1.1). It combines commercially available and clinically approved modules – the tracking camera, the surgical engine which powers the tool, and the foot pedal – with novel software and hardware modules, namely the NC software running on a laptop PC and the NC unit, a micro-controller utilized as central A/D interface for all devices.

Intraoperative Navigated 3D Ultrasound

The localization of vessels and the determination of the tumor boundary are two critical tasks during brain tumor surgery. In image guided neurosurgery these structures are identified with the means of radiology images and navigation systems or stereotactic frames. In ENT, maxillofacial, and orthopedic surgery, one usually finds a sufficient amount of bony, i.e. rigid, landmarks to determine a target position which is then localized with a tracked pointer. In brain surgery, the accuracy of the spatial mapping between patient space and image space with the means of tracked body markers is impaired by the fact that the brain is a soft organ. As an effect of CSF outflow and tissue resection, the target and risk structures are shifted.

One means to compensate for this effect is intraoperative imaging. Different imaging technologies are utilized intraoperatively in neurosurgery: X-ray angiography, MRI, or CT ultrasound are the most common, but also the usage of functional modalities like

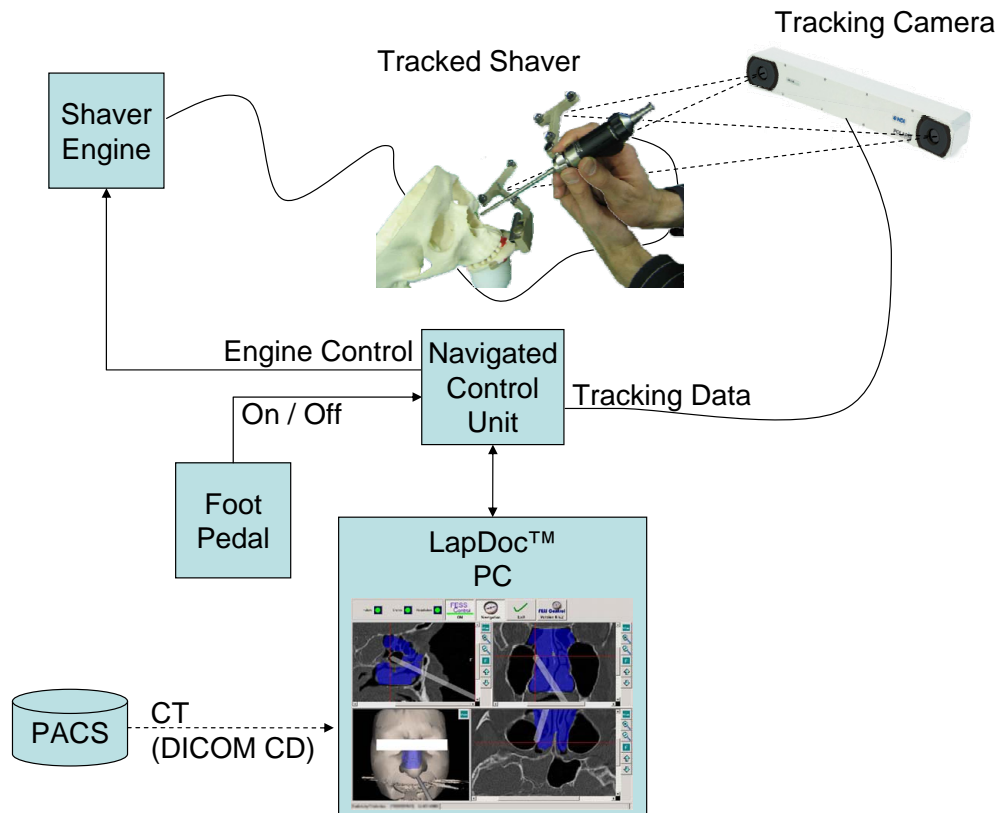


Figure 1.1: NC FESS architecture.

PET has been reported [Jabbour *et al.*, 2009; Vitaz *et al.*, 1999; Okudera, 2000; Knauth *et al.*, 1999; Kubota *et al.*, 2004; Chandler *et al.*, 1982; Rygh *et al.*, 2006; Ballanger *et al.*, 2009].

The LOCALITE BrainNavigator integrates intraoperative ultrasound images with pre-operative MR images. A binocular tracking camera and attached body markers are employed to spatially register an MR image stack, the patient, a sterile pointer, and a tracked ultrasound probe. A volumetric US image is reconstructed from the 2D US slices which are acquired with the tracked probe. This image is rigidly registered with the MR image and both images are visualized in a three-planar sliced representation which shows both modalities in a semitransparent overlay. By identifying landmarks in both images, brain shift can be estimated by the surgeon [Lindner *et al.*, 2003].

The components of the LOCALITE BrainNavigator and their connections are presented in Figure 1.2. The setup includes a commercially available NDI Polaris® tracking camera and a Siemens ultrasound device. Both systems are attached to a central PC which runs software for volume reconstruction, patient-image registration, and visualization. The tracking system is attached through a proprietary interface. The US images are captured from the S-Video output of the device. The latter is in principle a standardized interface which would be accessible in most ultrasound devices. The captured frames show the complete screen layout of the ultrasound image rather than

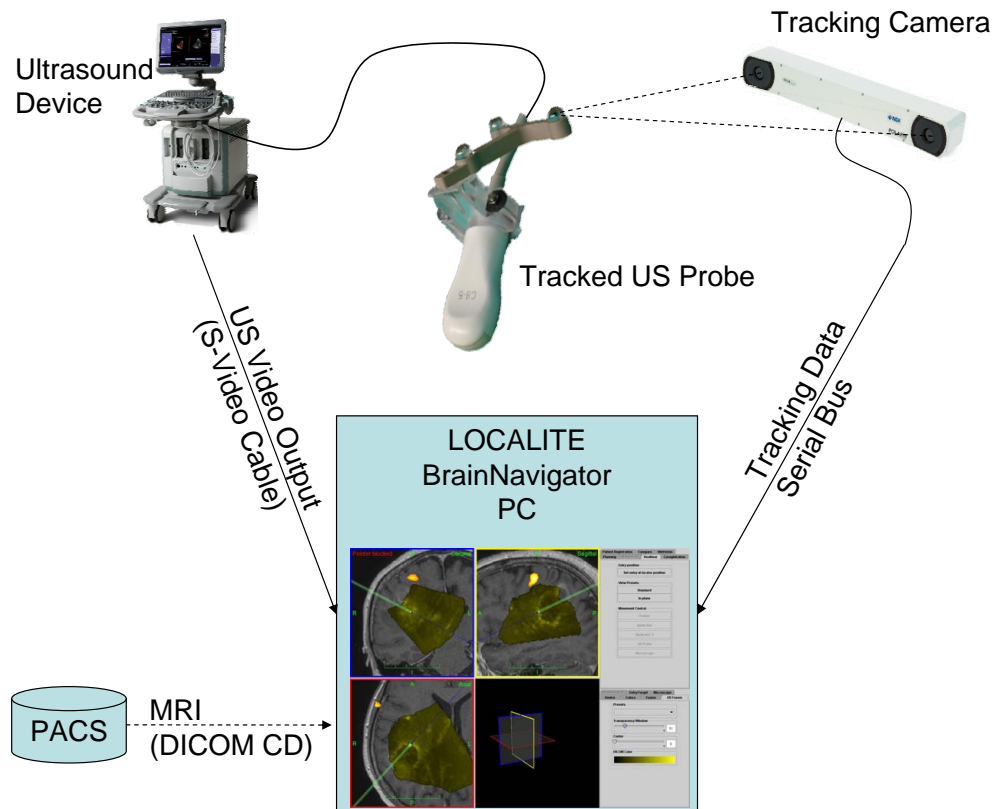


Figure 1.2: LOCALITE BrainNavigator architecture.

only the ultrasound image. In order to crop the correct region from the frames, the BrainNavigator needs to be adapted to the specific layout of the attached device.

Discussion

The NC FESS and the BrainNavigator show similar system architectures. Both setups contain a commercially available tracking system. Both systems access the navigation data through a proprietary interface which makes it impossible to run either of the systems with a tracking system of a different type than they were designed with. For a health care provider who already has tracking systems of the "wrong" type equipped in the operating rooms, this limitation results in the requirement to invest into a second line of systems.

The unavailability of appropriate technical standards for data exchange in such systems makes it impossible to re-use or exchange any of the contained modules. On all levels, from the control data exchanged in the NC system to the streaming ultrasound video, technical standards would be required in order to transform these systems into open architectures.

The NC FESS system is not networked with the PACS for the import of patient images and for the archival of planning data. It imports DICOM images for intervention planning and intraoperative image guidance from storage media. The planning software

and intraoperative assistance software are integrated into one monolithic application. It is not possible to import segmentation results which were generated with different software. The exchange of planning information and patient images between two NC laptops is only possible via storage media (CD-R, DVD-R, or flash disks).

The Localite system can be attached to a PACS network for image import through the network. In the commercially available version of the system, the export of the reconstructed ultrasound volume to a PACS server is not supported. A DICOM standard for volumetric ultrasound has just recently been released in 2008. Localite did evaluate the export of the ultrasound images as DICOM 3D US datasets in a joint project with ICCAS in an experimental setup of an integrated OR environment in Leipzig.

Robotic Assistance in Endoscopic Surgery

Falk et al. [Falk *et al.*, 1999, 2000] presented an approach for totally endoscopic coronary bypass graft anastomosis. This task, the surgical connection of an arterial bypass with a coronary artery, requires a high level of dexterity. With regular endoscopic tools – rigid lengthy devices which are introduced through incisions on the chest – suturing the vessels was described as almost impossible [Stevens *et al.*, 1996; Mack *et al.*, 1997]. The anastomosis technique proposed by Falk et al. uses the da Vinci[®] telemanipulator. The da Vinci[®] is a robotic device which is used to steer surgical manipulators with articulated end effectors [Ballantyne & Moll, 2003]. It consists of 3 modules:

- The master console contains a binocular video display for stereoscopic endoscopy and two handles with which a surgeon controls the motion of the surgical instruments on the slave unit.
- The slave module is a robotic manipulator with up to three instrument arms which can be equipped with articulated tools and an extra arm which holds a stereo endoscope.
- The control unit translates the user input at the master unit into control signals for the slave unit. Signal processing algorithms can be employed to filter out tremor and for motion scaling up to a degree of 5 to 1.

With military applications of telesurgery in mind [Satava, 2003], the da Vinci[®] system was developed as a modular system from the beginning. Its modular architecture facilitates setups, where the master and control unit are connected via a satellite or landline connection. In several research projects, the modularity of the system was "alienated" for the introduction of additional modules into the setup.

Voruganti et al. presented an augmented reality setup with the da Vinci[®] system for intraoperative assistance in the localization of coronary arteries [Voruganti *et al.*, 2007]. Access to the endoscopy video signal is possible since the images are transferred via standard S-Video cables. Voruganti et al. used a standard frame grabber card to digitalize the video signal into a PC, where they employed augmented reality technology



Figure 1.3: da Vinci[®] surgical system consisting of master console (left), control module (right), and slave unit (center). The image on the bottom right shows an articulated instrument for the da Vinci[®]. (Images Courtesy of Intuitive Surgical, Inc., Sunnyvale CA, USA)

to enhance the images with overlaid visualizations of a patient specific surface model of the coronary arteries (see Figure 1.4). The augmented frames were output via the S-Video socket of the graphics card to the da Vinci[®]'s control unit.

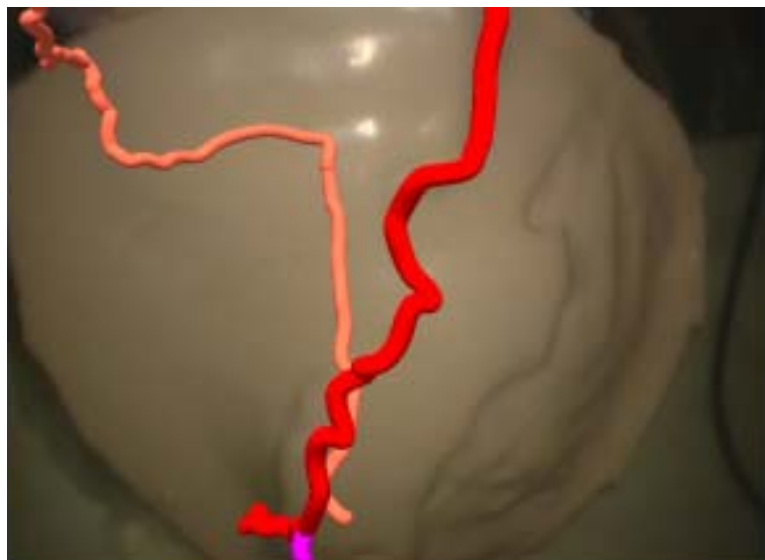


Figure 1.4: Augmented reality in the da Vinci[®] master console: preoperatively segmented coronary arteries are superimposed to the endoscopy frames [Trautwein *et al.*, 2009].

Leven et al. [Leven *et al.*, 2005] used a proprietary API which is available from Intuitive Surgical for limited access to the da Vinci[®]'s control module. They presented a setup where a miniature US probe is mounted onto one of the da Vinci[®]'s arms and can be operated inside the patient to acquire ultrasound images of the internal organs. Optical tracking technology is used to estimate the relative position and orientation between the endoscope and the ultrasound probe. The ultrasound image is superimposed onto the stereo endoscopy images in real time so that it appears as a fan on the tip of the probe.

Discussion

The setup by Voruganti et al. demonstrated the flexibility, a modular system design in CAS gains from standardized interfaces. In the case of the standardized video interface, they were able to interpose their AR module between the camera and the console without the necessity for any adaptation of the da Vinci[®] modules.

Leven et al. use a proprietary interface to access a limited set of functionalities which was made accessible for external modules by Intuitive Surgical. The proprietary nature of this API restricts the usage of the Canvas system to one telemanipulator. In order to integrate the device with a different device, e.g. the ZEUS system, a re-design of the interface between the US module and the controller of the telemanipulator is necessary.

1.2.2 Summary

The technical development of mechatronic devices and image processing methods has lead to very far developed CAS systems in the past 30 years. The introduction of these systems, especially the increased amount of imaging modalities which are utilized in the preparation and conduction of surgery, has lead to an overhead of workload which arises from the operation of these devices. Modular architectures are not uncommon in CAS. The workflow in modern ORs is hampered by the lack of standardization of interfaces for integration the modules of CAS systems in the OR and by the lack of standardized interfaces between CAS devices and information systems in hospitals.

1.3 Aim of the Thesis

The aim of the thesis is to specify interfaces for distributed surgical planning systems and modular intraoperative assistance systems based on open standards or specifications. This aim is reached by pursuing the following goals:

Firstly, the interfaces between preoperative planning and intraoperative system will be investigated. The requirements will be analyzed which would have to be met in an attempt to specify an open standard for the transfer of planning results from preoperatively applied planning systems to intraoperatively employed displays and assistance systems.

Secondly, particular interest will be paid to the question whether the DICOM standard can be used or extended to comply with these requirements. Use cases will be identified for which this is the case and DICOM extensions will be proposed for planning data exchange in these use cases.

Thirdly, the intraoperative requirements to an open interface for modular CAS systems will be investigated and existing open standards and protocols will be identified which comply with these requirements.

Forthly, a "surgical middleware" will be specified and implemented on the basis of these standards and protocols which enables the flexible integration of CAS modules.

Finally, the feasibility of the chosen approaches and the proposed interfaces is tested. Therefore, modular CAS systems are developed which utilize these interfaces.

1.4 Structure of the Thesis

In Chapter 2, the historical development and actual state of the art in computer assisted surgery in general and in surgical informatics in particular is presented and discussed in order to clarify the theoretical and practical foundation on which the analysis, specifications, and implementations which are presented are based upon in the thesis is based.

In Chapter 3 the requirements to an infrastructure for transfer of planning information into the OR and to an infrastructure for communication in an intraoperative distributed system are investigated. On the example of five interventions from cardiac, orthopedic, ENT, and neurosurgery it is shown how surgical workflows (see Section 2.1.3.2) can be utilized to derive these requirements. Also in Chapter 3, the differences in the requirements between the preoperative use cases and the intraoperative use cases is investigated. In Chapter 4 the applicability of DICOM for the transfer and storage of surgical planning data is discussed and two DICOM extensions are presented which were specified in this regard. Chapter 5 describes the TiCoLi, a software library for intraoperative data exchange and network management which is based on open standards and open source software.

The operability of the specifications presented in Chapter 4 and 5 is proven on two examples in Chapter 6. A system for preoperative planning and intraoperative assistance of transapical aortic valve implantation and a system for preoperative and intraoperative modeling of the central sulcus for brain tumor surgery are described. Both systems utilize the proposed DICOM extensions for storage and transfer of planning data. The intraoperative modeling system in the second example is integrated with the TiCoLi. In Chapter 7, a summary of the results is given and the presented specifications and implementations are discussed with regard to the aim of the thesis. In the outlook in Chapter 7, recommendations for future steps in the direction of standardized interfaces for CAS systems are made.

Chapter 2

State of the Art

Computers can be found in every aspect of modern hospital work. In the domain of surgery, the predominant applications for computers are Surgical Information Systems (SIS) which help organizing patients, staff, equipment, rooms, and other facilities and Computer Assisted Surgery (CAS) systems which support surgeons in preparation or conduction of surgical interventions. The SIS is usually part of a Hospital Information System (HIS) through which patient data, billing information, and other information can be exchanged between departments. CAS systems, in contrast, are usually "islands of information" [Lemke & Berliner, 2008] which require manual input and output of information. The state of the art regarding the interconnectivity of CAS systems and the integration of CAS systems to SIS and other information systems is presented in Section 2.2. Beforehand, an introduction into the field of CAS is provided through a short overview of the historic development and current technologies of CAS.

2.1 A Brief History of Computer Assisted Surgery

The utilization of computers in surgery goes back to the mid of the last century when computers were introduced for the monitoring of vital signs in intensive care units and during surgery. Exactly when and by whom the term "Computer Assisted Surgery" (CAS) was coined is unclear but the event is often related to the development of surgical navigation systems in the 1970s [Schlöndorff, 1998]. Today, CAS subsumes a variety of techniques for treatment planning and intraoperative assistance. In this section, the historical development of certain aspects of CAS is presented without intending to be exhaustive.

2.1.1 Image Guided Surgery

Visual perception is one of the limiting factors of surgery [Jolesz, 1997]. During an intervention, the surgeon's natural view ends at the exposed surface, while the surgical targets usually lie beneath the exposed surfaces of the body or the exposed internal organs. Furthermore, the incision through which an intervention is performed physically

constrains the field of view, limiting the amount of anatomical reference points which could facilitate identification of structures. The latter is intensified by the forthcoming of minimally invasive surgery. Miniaturization of tools, progress in endoscope and port technology and the development of new microsurgical devices lead to a continuous decrease of the incisions through which surgery can be performed and thereby to increasing limitations to the field of view. The execution of surgical maneuvers through the proverbial "key hole" is aggravated by the limited degrees of freedom offered by endoscopic tools and due to the leverage effect. [Scott-Conner & Berci, 1993].

Image Guided Surgery (IGS) facilitates perception with the means of imaging systems. The history of IGS begins decades before computers were invented. Surgical planning based on a radiological patient image was reported for the first time on January 14, 1896 in Birmingham, when a needle was removed from a ladies hand with the help of a radiograph – 9 days after Wilhelm Conrad Röntgen's original publication "*Über eine neue Art von Strahlen*" ("On a new kind of rays") [Röntgen, 1895; Mould, 1980]. Röntgen and his followers had transformed the fundamentals of medicine in general and surgery in particular literally over night – the importance of medical imaging in diagnosis, treatment, and follow up control is increasing ever since. Today, preoperative and intraoperative images are utilized in the preparation and/or conduction of almost every surgical non-emergency intervention. A multitude of modalities including endoscopy, intraoperative X-ray, MR, or ultrasound imaging are used to acquire 2D or 3D images to extend the surgical perception beyond the limits of direct visual perception.

The following paragraph summarizes the development of the imaging technology which is available for diagnosis, treatment planning, and intraoperative assistance in a 21st century hospital.

Diagnostic and Interventional Imaging – Past to Present

The first case of a surgical intervention which was performed under direct guidance of a technically acquired patient image was reported in 1869 when Pantaleoni used a modified cytoscope to cauterize hemorrhagic uterine growth [Nezhat, 2005]. Endoscopic techniques have since then been developed for every surgical discipline, including among others, laparoscopic approaches in pelvic and abdominal surgery, arthroscopic interventions in joints, and bronchoscopic procedures in the airways. In the late 1970s, endoscopes were first equipped with video cameras, allowing clinicians to observe the intervention on a screen rather than through an optical lens on the scope. The immediate advantage of video endoscopy and the reason for its invention was that it removed many of the ergonomic limitations from endoscopy and was an enabling technology for complex interventions [Nezhat *et al.*, 1986]. A side effect of video endoscopy was that once the endoscopy images were available as electronic signals, it was possible to use (analog and later digital) computation methods to enhance images, extract structures from images (e.g. [Vogt *et al.*, 2005; Kaufman & Wang, 2002]), and augment images (e.g. [Kawamata *et al.*, 2002; Bockhold *et al.*, 2003; Vosburg & San José Estépar, 2007]).

Clinical radiology emerged in the late 19th century and early 20th century. Röntgen's

discovery, X-rays, and the invention of X-ray imaging techniques enabled physicians to see behind the surface of patients and lead to a tremendous progress in clinical diagnosis of fractures and diseases such as, e.g., cancer, tuberculosis and rickets. For surgeons, radiographs quickly became a standard means in preoperative intervention planning and in-situ orientation.

In 1971, Godfrey Newbold Hounsfield invented the CT scanner, an invention for which he was awarded the Nobel price in 1979. The pervasion of this technology enabled diagnosis and planning based on 3D information rather than projective images. In the 1980s, methods from Computer Assisted Design, and Computer Assisted Manufacturing (CAD/CAM) were applied for extraction of organs from CT (and later MR and Ultrasound) images and visualization of the resulting 3D structures.

Besides X-ray images, a multitude of imaging modalities was introduced to medicine during the last century. In 1940, Gohr and Wedekind investigated the applicability of ultrasound for medical diagnostics [Gohr & Wedekind, 1940]. The first hand-held B-mode ultrasound scanners became available on the market in the late 1960s. 3D Ultrasound became available in the 1980s. In the late 1970s, the first articles were published on the application of ultrasound as a real time imaging modality for biopsy needle guidance in a Russian gynecology journal [Kazi *et al.*, 1979], marking the beginning of ultrasound guided surgery. Around the same time, the first report on the application of Doppler ultrasound for the visualization of blood flow was reported by Stevenson, Brandestini *et al.* [Stevenson *et al.*, 1979; Brandestini *et al.*, 1979]. Invention of Intravascular ultrasound (IVUS) in the late 1980s [Yock *et al.*, 1989] enabled imaging of vascular stenosis and other cardiovascular morbidities in a hitherto unavailable quality and without the requirement for X-ray contrast agent. During cardiovascular interventions, IVUS imaging is used to monitor and control the process of, e.g., vascular stent deployment [Nakamura *et al.*, 1994].

In 1973, Paul C. Lauterbur invented Magnetic Resonance Imaging (MRI) [Lauterbur, 1973]. The first MR images of human anatomy were published by Hinshaw *et al.* in 1977 [Hinshaw *et al.*, 1977] and the first acquisition of 3D MR images was reported in 1982 [Muller *et al.*, 1982]. Compared to CT, MR has a far better sensitivity for soft tissue, making it a powerful tool in the diagnosis of diseases of the internal organs and brain. This stands against inferior bone contrast, less temporal and spatial resolution and spatial accuracy. Early applications of MRI for treatment planning are, e.g., Kim and Wineberg [Kim & Weinberg, 1986] in spinal surgery, Kangarloo *et al.* in renal tumor surgery [Kangarloo *et al.*, 1986], Goldman *et al.* [Goldman *et al.*, 1986] in vascular surgery, and Fasano *et al.* [Fasano *et al.*, 1986] in brain tumor treatment. In IGS, MR was first used in planning of tumor resection, especially for liver and brain tumors. The excellent soft tissue contrast and the fact that MR imaging has no harmful effect on the patient or the clinical personnel lead to the development of intraoperative MRI systems.

Nuclear medicine imaging techniques are based on the introduction of radioactive agents into the patient. The radioactive isotope (tracer) is injected as part of a biologically active molecule. A gamma-ray sensitive scanner is rotated around the patient,

similar to the X-ray detector in a CT scanner, and a volumetric image of the spatial distribution of the radioactive agent is created. By choice of the bioactive molecule, different physiologic or pathologic effects or processes can be visualized through this technique. Fludeoxyglucose (FDG), e.g., is a biochemical analog to glucose and will be found with a high concentration in regions of high metabolic activity. FDG with a Fluorine-18 tracer is commonly used in the acquisition of diagnostic Positron Emission Tomography (PET) images in tumor diagnostics. The concept of positron emission imaging was introduced in the 1950s by David E. Kuhl and Roy Edwards. The concept was first applied to brain tumor detection by Brownell and Sweet at Massachusetts General Hospital in 1953 [Brownell & Sweet, 1953] to acquire projective 2D images of metabolic activity in brain tumor diagnosis. In the early 1970s, David Chesler was the first to successfully advance this technique to tomographic imaging [Chesler, 1971]. Today, PET and SPECT as well as 2D scintigraphy are established in clinical routine for the depiction of metabolic processes [Abramyuk *et al.*, 2008].

The combination of two or more imaging modalities in order to combine their advantages or to compensate for their disadvantages is known as multimodal imaging. In order to correlate the image information from different modalities, an accurate coregistration of the coordinate systems of both images or volumes is required [Staemmler *et al.*, 1995]. There exist two principal approaches to this issue:

- The scanning procedure is taken out in a manner which results in a known relation between the image frames. For this purpose, multimodal scanners, where two modalities, e.g. CT and PET, are integrated into one device are available [Gong *et al.*, 2005]. Other approaches utilize tracking technology or patient rests on which the patient is fixated and moved from one scanner to the other to obtain a spatial relation between the isocenters of the imaging devices and patient which is then used to coregister the images [Lindner *et al.*, 2003].
- The images are acquired independently and are coregistered based on landmarks or otherwise defined similarity measures [Maintz & Viergever, 1998; Zitova & Flusser, 2003]. To facilitate the process of image coregistration, the usage of body markers or stereotactic frames has been proposed [Hemm *et al.*, 2005].

Multimodal imaging is to a large extent dependent on the integration of different modality images. The advance of multimodal image analysis in treatment planning is one of the origins of the demand for the integration of devices for imaging, patient modeling, treatment planning, and computer assisted surgery into one comprehensive information system [Mösges, 1993; Staemmler *et al.*, 1995].

Stereotaxy and Surgical Navigation

Stereotaxy and Surgical Navigation are techniques which aim at correlating the physical space in which surgery takes place (in the following referred to as patient space) with the virtual space of 3D images [Gildenberg *et al.*, 2009]. Stereotaxy was first applied in animal experiments in 1908 by Sir Victor Horsley and Robert H. Clarke

at University College London Hospital, UK. The device consisted of a probe holder which was mounted in a Cartesian frame which was fixated on the specimens head. The frame was used to exactly measure the location of landmarks on the specimen's skull in three dimensions. Stereotactic atlases related the position of the landmarks to target spots inside the skull. The inter-subject variability of cranial landmarks impaired the accuracy of early stereotaxy devices. The first stereotactic device which was routinely used in human neurosurgery was invented in 1947 [Spiegel *et al.*, 1947]. Similar to the Horsley-Clarke apparatus this device based on a stereotactic atlas. Instead of cranial landmarks, this atlas consisted of photographic coronal slices of the brain and localization of target points based on landmarks in the ventricular system. This led to higher accuracy in sub-cortical interventions. Despite these incremental improvements, the problem of inter-subject variations in the landmarks remained to be unsolved. From 1949, Jean Talairach experimented with a stereotactic system which included a radiographic device. X-ray images were acquired with a rectangular wire grid in place and were used as patient-specific stereotactic atlases. This marks the beginning of patient image guided stereotaxy. In 1978, Russel A. Brown was the first to develop a system which used CT images to guide stereotactic surgery. Today, the Brown-Robbers-Wells stereotactic system and similar systems are widely used in neurosurgery, ENT surgery, radio therapy, and other applications [Galloway & Peters, 2008]. MR images are applied for stereotaxy in brain surgery since the late 1980s [Peters *et al.*, 1987; Pillay *et al.*, 1992; Vindlacheruvu *et al.*, 1999]. For multimodal stereotaxy, special markers were developed which are visible in CT and MR images and can therefore be used for inter-modality registration [Alesch, 1994; Cosman & Roberts, 2002].

Surgical navigation systems, or frameless stereotaxy systems, apply various measurement methods to determine the position of a surgical tool or pointing device and relate it to coordinates in patient images. Roberts, Friets *et al.* were the first to track an OR microscope using an acoustic system [Roberts *et al.*, 1989; Friets *et al.*, 1989]. Kosugi, Watanabe *et al.*, utilized an articulated arm to acquire landmark positions and monitor tool motion in the OR [Watanabe *et al.*, 1987; Kosugi *et al.*, 1988]. Both groups utilized the acquired spatial information to drive visualization of CT images. Originating in neurosurgery, frameless stereotaxy was introduced to ENT surgery in 1990 [Adams *et al.*, 1990]. Several design-inherent limitations of articulated arm systems motivated the development of contactless systems with free-hand or handheld localizers. In 1993, the sonic tracking concept which was already used by Roberts *et al.* to track an OR microscope was applied to tracking of a small handheld probe [Barnett *et al.*, 1993; Reinhardt *et al.*, 1993]. To overcome accuracy-problems of acoustical tracking systems, optical tracking systems were developed with active infrared LED markers [Zamorano *et al.*, 1994; Tebo *et al.*, 1996], passive IR reflecting markers [Wiles *et al.*, 2006], or passive markers with patterns in the visible light spectrum [Balachandran *et al.*, 2006]. Optical tracking systems require uninterrupted lines of sight between the markers and the tracking cameras. Experiments with magnetic tracking systems which do not require a line of sight began in 1994 [Manwaring *et al.*, 1994]. The sensibility of magnetic tracking systems to interfering magnetic fields as they are generated by

moving metallic objects retarded the pervasion of magnetic tracking technology in applications, where optical systems are applicable in principle. Fields of application for magnetic tracking are minimally invasive techniques with flexible needles or catheters in interventional radiology, biopsy [Zhang *et al.*, 2006] or endocardial interventions [Faddis *et al.*, 2002].

2.1.2 Preoperative Planning

Besides diagnostic applications, the instantaneous effect of the intervention of radiography and other imaging techniques was the possibility to plan surgery beforehand. Today, there is no surgical discipline where image based intervention planning is not part of good clinical practice standards. An overview of the employment of the different imaging modalities in therapy planning for surgery is presented by Vannier *et al.* [Vannier *et al.*, 1996].

In many surgical disciplines, including orthopedic surgery, trauma surgery, but also in spinal and dental surgery, the standard method for treatment planning used to be based on hard-copy 2D radiographs or tomographies on light boxes. In the early 1990s, the supersession of hardcopies by digital radiology image management gave rise to digital planning applications in these disciplines. The increasing demand for 3D visualizations and 3D modeling, especially in applications which involved stereotaxy or navigation, made computer assisted planning inevitable [Kettenbach *et al.*, 1997].

Kikinis *et al.* give an introduction into techniques and applications in computer-assisted surgical planning with 3D reconstructions [Kikinis *et al.*, 1996]. The focus of this article lies on the preoperative visualization of the anatomy of certain brain structures. Other typical applications in computer assisted surgery planning are:

- **Implant selection:** The optimal type, size, and positioning of one or a set of implants is determined in a patient-specific manner based on radiology images. In computer-assisted implantation planning, the selection is performed based on radiology images and digital implant templates. Implantation planning can subsume the determination of bone or other tissue which need to be resected in order to place the implant (see below under resection planning). The advantages and disadvantages of digital implant planning have been investigated in clinical study by Winter *et al.* in the scope of the SaxTeleMed program [Winter *et al.*, 2002].
- **Access planning:** The access to a surgical target, e.g. a lesion, is planned in order to minimize the sometimes inevitable paresis induced by creating a passage through healthy tissue. An example for such an approach was presented by Vaillant *et al.* in [Vaillant *et al.*, 1997]. Another aspect of access planning is the preoperative determination of a target position and orientation for, e.g., a biopsy needle in navigated or stereotactic percutaneous interventions (e.g. [Mozer *et al.*, 2006]).

- Path planning: The access to a situs which can be reached by following the natural cavities of the human body (the respiratory or digestive system or the blood vessels) is planned. As an example in this respect, a system for cardiovascular path planning is commercially available from 3mensio Medical Imaging. It calculates the endovascular diameters and distribution of stenotic processes along the arterial access to the ascending aorta starting from two entry points on the left and right femoral artery. This information supports the surgeon in the decision for one side to enter the vascular system and in the selection of a catheter.
- Target Region of Treatment (TROT) planning: The region of the patient's body in which a treatment shall be applied is determined. The definition of a workspace, for example in Navigated Control (see Section 1.2.1), or the determination of energy targets in radiotherapy or interventional radiology fall into this category.
- Resection Planning: The volume and exact boundary of tissue is planned which will be resected during surgery. In the case of lesions or focuses of infection, planning is usually a trade-off between resection of as much malicious tissue as possible while sparing nearby risk structures or, in general, healthy tissue. An advanced computer-assisted planning tool which includes resection planning for brain tumor surgery was presented by the group of Helmholtz-Institute for Biomedical Engineering at RWTH Aachen [Bast *et al.*, 2006].

All of the presented techniques and approaches have in common, that they include the extraction of higher-level information (predominantly geometric and textual information) from image data. With the help of image processing algorithms, a surgeon, radiologist, or other clinically trained person generates a *model* of the patient. The model contains numerical measurements, geometric descriptions of structures and trajectories, and other forms of patient specific knowledge which are relevant for the specific planning task. For various applications, the information which can be directly obtained from medical images is insufficient for determination of an optimal strategy. Especially in cases where operating on soft tissue is involved, the physical simulation of the outcome is required.

The integration of the planning tools and their results into a network infrastructure and a central repository requires data structures to describe all these kinds of information. Compared to radiology image databases in PACS, a database for surgical planning results is required to be very versatile regarding the data it contains. Besides images, data structures are required for geometry, for descriptions of an intended course of action (called surgical workflow), for simulation results and generic models and patient-specific parameters on which simulation was performed, OR setup descriptions, lists of required equipment and consumables, and other kinds of data.

2.1.3 A New Paradigm: Model Guided Therapy

It is difficult to define by whom and when MGT was invented. Following the definition of a model as presented by philosophers like Stachowiak [Stachowiak, 1973] or Wüstenneck, every abstraction of an entity, which a second entity derives or defines in order to facilitate understanding of or control over the first entity, is a model. From this perspective, MGS is as old as surgery itself, when considering early anatomical drawings and the *mental models* surgeons construct from diagnostic data, images, and in-situ findings. Modeling, especially shape modeling and modeling of biomechanics, has always played a role in CAS. Early approaches in CAS incorporated the idea of a model of the patient that goes beyond images already in the 1980s [Lemke, 1985].

The concept of modeling was introduced to computer assisted radiology and surgery in the context of modeling the geometry of organs, lesions, or other anatomically, physiologically, or pathologically interesting structures from tomography images. Segmentation, i.e. the extraction of structures from images, and the extraction of surfaces which envelope these structures were for a long time synonym to the term patient modeling. For an overview over segmentation and surface extraction methods, the reader is referred to the literature, e.g. [Pham *et al.*, 2000; Ayache *et al.*, 1996; Rogowska, 2000]

Multimodal imaging (see above) can be seen as a progenitor of the concept of Model Guided Therapy (MGT). In 1994, Lemke *et al.* described the concept of a multimedia medical workstation with access to text, sound, graphical (1D, 2D, 3D), and other information [Lemke *et al.*, 1994]. The proposed concept relies on providing a hypertext-like relation between multimodal patient data with the aim to facilitate navigation through a patient file. The HYPERMED system which was created to proof this concept enables the user to define, alter, and delete links between patient media. The system contains a mechanism which automatically establishes connections. The HYPERMED terminal can be used to access and browse through the data and as a teleconferencing system for virtual cooperation.

In 2000, the "Modeling & Simulation in Medicine: Towards an Integrated Framework" workshop was held at the United States National Library of Medicine. The goal of the meeting was to formulate a ten-year vision for research on modeling and simulation in medicine [Higgins *et al.*, 2001]. In the center of this vision stands the development of an atlas of the human body which encompasses all physical scales, from molecular processes on the nano-level up to the shape and function of organs and complete organisms. This atlas, the *Integrated Digital Human*, was seen as a potential basis for predictive modeling (simulation) of biological processes, for better therapeutic assistance based on the fusion of intraoperative data with models, as a gold-standard for validation of simulation systems, for the creation of what was called a "body-double" which serves a repository for multimodal patient information, and as a basis for high-fidelity medical training simulators.

In the same year, the Physiome project (<http://www.physiome.org/>) picked up this idea by founding an open database for the collection of models of biologi-

cal processes on all scales and from multiple sciences, including biology, chemistry, physics, and medicine [Bassingthwaite, 2000].

In 2001, Shahidi et al. published a white paper which summarized their perspective on the "challenges and opportunities" in the CAS research [Shahidi *et al.*, 2001]. As the first among seven problems which require to be solved they name the development of techniques for generating Patient Specific Models (PSM) as a basis for treatment planning and simulation.

In a 2008 publication, Lemke and Berliner presented a formal definition of the term "Model Guided Therapy":

"Model Guided Therapy (MGT) is a methodology complementing Image Guided Therapy (IGT) with additional vital patient-specific data. It brings patient treatment closer to achieving more precise diagnosis, more accurate assessment of prognosis, and a more individualized planning, execution and validation of a specific therapy."

[Lemke & Berliner, 2008]

By Lemke's definition, MGT is founded on two models:

- a Patient Specific Model (PSM) and
- a Situation Model (SM).

2.1.3.1 The Patient Specific Model

The PSM is a comprehensive representation of all available information about one patient. The PSM can be based on generic Patient Models (gPMs). gPMs are not patient-specific but represent universal knowledge about aspects of the human organism. For example, a gPM could represent the *average shape* of the human heart and possible deviations from this average. During modeling, patient specific data can be utilized to adapt a gPM to the specifics of one patient. The resulting concrete expression of the abstract gPM is called PSM.

The PSM integrates input from various data sources, including textual information from the patient record or other documents, 1D signals from sensors, 2D images, 3D volumes, 3D surface models and higher-dimensional data, for instance already registered multimodal images or 3D+t data from real-time volumetric imaging modalities. Lemke lists the following modes of a potential PSM:

- Macroscopic morphology (shape of organs, tissues)
- Microscopic morphology (fiber directions, ...)
- Deformation
- Cardiovascular state (blood pressure, heart rate, oxygenation)
- Demographic information (age, weight, ...)
- Diagnosed diseases

- Known genetic disposition
- Known allergies
- Applied medication, distribution of drugs in the body
- ...

To construct a PSM, gPMs which contain knowledge from the following disciplines can be required [[Lemke & Berliner, 2008](#)]:

- Anatomy
- Physiology
- Biophysics
- (Bio-) Chemistry
- Pharmacodynamics
- Pharmacokinetics
- Genomics
- Proteomics
- ...

An MGT system has to be able to collect this data from preoperative and intraoperative data sources, temporally and spatially coregister it, and generate a PSM by fusing the data with gPMs. Lemke points out that in order to handle the uncertainties and inconsistencies which are in the clinical environment inevitably gained from incomplete and noisy data, the PSM is required to be based on probabilistic methods, such as Bayesian modeling [[Lemke & Berliner, 2008](#)].

The development of one monolithic system apt to process the diverse kinds of input data and implements the manifold gPMs would require an enormous effort, the involvement of experts in all the aforementioned field from Biomedicine as well as Computer Sciences. Such an attempt, if it were successful, would lead to a huge soft- and hardware system which would be difficult to customize, install, maintain and extend. In an MGT system, the construction and maintenance of the PSM can only be performed in a cooperative manner by distributed systems.

Patient Model Projects

There are several large-scale projects on a European or international level which aim at the definition of multimodal models for utilization in simulation of physiologic and pathologic processes and to facilitate treatment planning and assistance.

The Visible Human project [NLM \[1989\]](#) was founded in 1989 by the US National Library of Health with the aim to create a full-body multi-modal atlas of human anatomy. In 1994, the original Visible Human dataset was released. It consisted of MRI, CT, and pathologic cross-sectional images of an adult male. One year later, the Visible Female dataset was released with similar images of an adult female. The Visible Human

datasets were used by many research projects as a reference database or gold standard for the development and evaluation of medical imaging methods.

The Physiome project [NSR , 2000] is an international effort to create a comprehensive description of physiologic processes. Physiome aims at creating a database which reflects the biological processes in natural organism on various levels of detail from the systemic view on the whole organism, down through the levels of organs, tissues, cells, and proteins to the level of genetic processes [Bassingthwaight, 2000]. The Physiome database is a growing list of mathematically and otherwise formulated models of biological processes on all of these levels. For the construction of a patient model, this collection can be seen as the building blocks of the gPM. In order to personalize the models from the Physiome gPM to construct a PSM, the parameters for each model need to be estimated for an individual patient.

The Physiome project emphasizes the usage of a common modeling and simulation tool, the JSim package. This facilitates the combination of multiple specific models into complex scenarios where the interaction between two processes on the same or on different levels of detail is regarded.

As of August 2009, the Physiome database contains 235 models. 33 research groups around the world are running projects which aim at extending the database or are using models from the database in simulation, treatment planning or other applications.

2.1.3.2 The Situation Model

The SM is a comprehensive representation of all available information about the operating room, the equipment and devices in the OR, the personnel, and the intervention itself. The situation model provides a basis for the interpretation of sensory input, and can be utilized by an integrated OR suite to adapt the environmental parameters and device settings in a situation-sensitive manner. Similar to the PSM, which is based on patient-specific data and generic patient models, the SM is based on intervention-specific status information and measurements as well as on generic models of a surgical procedure, called surgical workflow. Surgical Workflows (S-WFs) contain the universal knowledge about what can potentially happen during one kind of intervention.

In [Lemke & Vannier, 2006], Lemke and Vannier describe a central "intelligence" of a modular system residing in the *workflow engine* which monitors all activities in the OR and thereby maintains the SM. The workflow engine monitors controls all technical processes inside the OR. According to the situation, the workflow engine would, e.g., decide what is visible on a central display at the OR table or adjust the brightness of the room lights.

Analysis of Surgical Workflows

Workflow modeling has its origins in manufacturing, where scientist like Frederick Winslow Taylor and Henry Gantt began to study the organization of material flow and the organization of work at the beginning of the 20th century. One hundred years

later, at the beginning of the 21st century, workflows have evolved to be one of the key instruments for modeling of business and productive processes in various fields of applications. Especially in complex scenarios, where several people add to the fulfillment of one goal or the processing of one piece of work, workflows are commonly accepted for planning of such processes as well as for monitoring the advance of work.

With the increasing workload and cost pressure which were felt by hospitals during the past decades, workflow techniques were employed as a means to identify room for improving efficiency. From the 1970s, medical laboratories were in the focus of intense investigations in this regard [Shulman, 1978; Fitzgerald & Swanson, 1992], followed by considerations on optimizing processes and personnel deployment in hospital pharmacies [Hanna, 1983] and the question how the structural design of hospitals affects the efficiency of operations [Skaggs, 1984]. Electronic medical records were introduced to patient data management in hospitals in the 1990s as a means to streamline processes in hospitals [Srinivasan *et al.*, 1984]. Around the same time, workflow management was discussed in radiology in the context of exchanging worklists between a radiology information system and the imaging modalities and diagnostic workstations in the Picture Archiving and Communication System (PACS) [Garfagni & Klipfel, 1995; Schrader *et al.*, 1997]. In 1993, a DICOM standard was released for the exchange of worklist items between workstations and imaging devices in PACS [NEMA, 1993].

In surgery, workflow analysis started as the analysis of perioperative processes around the turn of the millennium. Delays induced by later arrival of patients, inflexible personnel and room scheduling, problems in the supply chains for tools and consumables and long turnover-times between interventions were identified as hindering influences to the throughput of the OR [Alon & Schüpfer, 1999; Geldner *et al.*, 2002]. Several strategies were proposed to reduce these issues, including the installation of an interdisciplinary OR management and the installation of an information system which facilitates the flow of patient data, OR schedules, personnel plans, and other organizational information [Riedl, 2002]. In 2005, Baumhove *et al.* presented evidence for the positive effect of OR reorganization on the employees' health, the case throughput and the economic efficiency [Baumhove & Schröter, 2005]. For the reduction of setup times and the time which is lost during surgery as an effect of difficulties to access data outside of the OR, the requirement for an integrated digital OR is identified (see Section 2.2).

Up to the mid of the 1990s, these efforts predominantly focussed on perioperative workflows and material flows. In the late 1990s, this focus was shifted by some groups towards the investigation of intraoperative workflows. Workflow models of intraoperative processes are created for three reasons:

- To analyze the efficiency of a whole process, an instrument, or the OR equipment and design.
- To create a universal model of a surgical procedure with the aim to create a

situation-aware CAS system, like it is by definition an integral part of an MGS system.

- To derive use cases for the specification of information systems and communication standards for OR integration (see Section 2.2.3).

In 2003, Warren Sandberg et al. published surgical workflows with a stronger focus on the intraoperative procedure course [Sandberg *et al.*, 2003]. The aim of this group was to derive requirements for the design of an integrated digital OR suite, the Operating Room of the Future (ORF), which was built at Massachusetts General Hospital (see Section 2.2.1). The workflows this group presented were strictly linear. While providing a good starting point for the requirements analysis for which they were created, they do not depict the flexibility of surgical procedures. In Section 2.2, their findings and similar findings reported by other groups regarding the inefficiency of intraoperative workflows due to limitations in the technical infrastructure will be discussed.

Situation Models based on Surgical Workflows

Research in the direction of situation-aware operating rooms or situation-aware CAS systems is in a very early state of development. The utilization of checklists as a means to increase safety in surgical procedures is one way to add situation awareness to the OR. In 2004, a US-based company called *Surgical Safety Institute* has begun to sell digital checklist systems which transfer the concept of standardized procedural safety checklists from aviation to surgery. Their systems require manual input according to a structured list of tasks from which a situation model is derived according to which decisions can be supported. Apparently, the device is not integrated with other OR devices, rendering it a passive monitoring tool rather than the workflow engine which was specified by Lemke et al. as the central intelligence which acts a director in the orchestrated interaction of modular CAS systems.

Lemke et al. pioneered into the direction of graphical representation of surgical workflows in the form of animated virtual OR scenes. They presented a rendering system which creates animated video sequences from abstract workflow descriptions [Lemke *et al.*, 2004]. The prior aim of these animations is the visualization of recorded workflows for analytic purposes. Nevertheless, the system has to be noted as one of the first if not the first to process a surgical workflow in order to drive a computational engine (in this case a rendering engine) with the contained information.

On the 2006 conference on Computer Assisted Radiology and Surgery (CARS), Ohnuma et al. presented a system for the intraoperative motion recognition of the surgeon's hand [Ohnuma *et al.*, 2006]. The aim of their research was the development of a robotic scrub nurse which analyzes the surgeon's hand motions and based on a workflow model is able to anticipate the tools which might be required by the surgeon in the near future. Their workflow model is based on probabilistic timed automata which are applied on large scale to describe the sequence of procedure steps as well as on a smaller scale where elemental movements of the surgeon are described as the basis for motion recognition. While highly specific to one particular use case and far

from generally applicable as a central controller, the proposed system shows some of the features which were defined as characteristics of an OR workflow engine.

At the same conference, Burgert, Neumuth, et al. presented their work on the fusion of workflow models with a knowledge base which is founded on the General Ontological Language (GOL) [Burgert *et al.*, 2006]. This link allows computational reasoning to be applied on surgical workflow models in order to compare or unify two workflows and in order to validate a workflow. This concept has the potential to be the basis for the data-driven generation of surgical workflows, where one probabilistic sequence of actions is generated from multiple concrete observations. The value of ontologies to describe the semantics of surgical procedures had before been investigated by Rossi Mori et al. [Rossi Mori *et al.*, 1997].

2.2 Surgical Informatics

"Surgical informatics is in a nascent phase as a discipline today. By definition, surgical informatics is the collection, storage/organization, retrieval, sharing, and rendering of biomedical information that is relevant to the care of the surgical patient. Its purpose is to seamlessly use computer-based informatics programs to provide comprehensive and decision making support to the health care team. As a result of applying surgical informatics to both usual and problematic surgical cases, improved decision making and problem solving in surgery are possible."

[Cleary & Kinsella, 2004]

Informatics has provided the theoretical insights and practical tools to structure and to integrate the handling of patient data and organisational content (schedules, billing, ...) in many aspects of healthcare. Hospitals are heterogeneous institutions subsuming several departments with differing requirements and preferences to the way data is handled and presented. In a modern Hospital Information System (HIS), all relevant information is archived in central digital repositories and can be exchanged between the departments [Prokosch & Dudeck, 1995]. Specialized subsystems implemented according to the specific needs of, e.g. radiology, laboratories, or surgical departments are connected to the HIS. The communication between the department is based on domain-unspecific standards such as HL7 (see Section 2.2.3). For Radiology Information Systems (RIS) and Laboratory Information Systems (LIS), specific standards exist (e.g. DICOM, LOINC, or VITAL) which better apply to the specifics of these domains. Projective mapping schemes exist, which "up cast" content from the specific domains to HL7 and other more general standards.

The analysis of perioperative workflows (see above) which was intensively performed since the 1980s in order to increase efficiency of surgical clinics, has revealed the need for an integrated Surgical Information System (SIS) [Alon & Schüpfer, 1999; Geldner *et al.*, 2002; Riedl, 2002; Sandberg *et al.*, 2003] The focus of the SIS developments

which lead to the systems which are commercially available at present lay on the organizational matters of the surgical workflow. SIS usually include modules for personnel and room planning, for OR scheduling, and for management of supply chains for material. HL7 can be utilized to exchange patient data or billing information between HIS and SIS.

The management of image data, patient models, and treatment plans which are handled during the preparation and conduction of surgery is not covered by these systems. Several publications and workshop reports addressed the ergonomic and workflow limitations induced by the increasing amount of high-tech devices in modern ORs which lack a common infrastructure.

In 1999 the "Technical Requirements for Image-Guided Spine Procedures" Workshop was held to determine the state of development for computer assisted spine surgery and to identify limitations and challenges. The results were published in a report which represents an excellent snapshot of the state of the art in computer assisted surgery [Cleary, 1999]. The clinical application in the focus of the workshop was spinal surgery, but many of the identified challenges can be regarded symptomatic for CAS in any surgical discipline.

In 2002, the NIH/NSF workshop on Image-guided Interventions recommended the development of an information system for the seamless integration of devices in the OR and the installation of a standardization body for development of the technical standards such an integrated system requires [Haller *et al.*, 2002]. The 2004 workshop "OR 2020 – The Operating Room of the Future" emphasized these recommendations and substantiated them to a higher level of detail. The focus of this workshop lay on the clinical and technical requirements for integration of CAS technology into operating rooms. The workshop report names five themes which were identified by the workshop participants as priority aims for the future development of CAS systems [Cleary & Kinsella, 2004]. Two of these themes regard the development of methods in image processing and robotics. The other three are related to the need for infrastructure and standards:

1. **Standards** *for devices and their use in the operating room (OR) are sorely needed. Every aspect of OR activity today is affected by their absence, from nonstandardized and incomplete patient records, to varied and unstandardized imaging formats of visual information that is needed during surgeries, [...]*
2. **Interoperability of devices** *is needed for development of a smoothly operating OR as well as for improved surgeries. Currently, most devices and computer systems function as stand-alone islands of information and their use requires a great deal of surgeons' time and effort. [...]*
5. **Communications issues** *must be addressed and aim toward attaining a common language, training requirements, and pro-*

ocols for effectively performing advanced surgeries and using telecommunications-ready tools as needed.

[Cleary & Kinsella, 2004], pages 3 – 4

The report gives five recommendations to future development in CAS, of which the first and the last relate to modularization and standardization:

1. *Standards, standards, standards. If there was an overarching theme of the workshop, this was it. [...]*
2. *Progress on the first recommendation will also enable progress on device interoperability. [...] A "plug and play" architecture for medical devices is also needed.*

[Cleary & Kinsella, 2004], page 4

The working group which focused on standards comes to the conclusion, that they are required on many levels: Surgically, computer assisted procedures require standards in the sense of common practices. On the technical side, the need for standardization is expressed in the workshop report on two different levels:

- On the application level, there is a need for standardized data formats in which systems can exchange medial data.
- On the network and transport level, a standard is required which enables plug-and-play functionalities in the OR.

In irregular intervals, these workshops reports document what the leading experts in the CAS community felt to be the key research topics and obstacles in pervading computer assistance in surgery. In the early 1990s, the emphasis was clearly on singular functionalities and technologies, like imaging, image analysis, or tracking. A decade later, many of the open issues regarding these technologies appear to be solved to a level where systems are clinically useful and topics like integration, interoperability, and modularization came to the fore.

As of 2005, the development of information systems, including data sources, processing and storing units as well as visual and other output devices, has reached an extremely high level. It is commonly agreed that the development of information systems in the field of CAS has produced technologies that have the potential to increase patient safety and outcome quality of surgery as well as facilitate education of young surgeons. In order to generate flexible system architectures which allow for the seamless integration of these systems into the OR and the information infrastructure of hospitals, progress in the field of communication systems for the OR are now required.

2.2.1 Integrated OR solutions and projects

The increasing urgency with which the demand for better system integration and improved workflows in the OR was submitted since the 1990s has produced a number of research projects and commercial products which offer solutions in this regard. Some of the most ambitious or successful attempts are presented in the following.

MedNet

The MedNet project established a digital network across seven hospitals associated with the University of Pittsburgh [Simon *et al.*, 1995]. MedNet connected ORs and intensive care units with diagnostic and research labs. It enabled the exchange of neurophysiological and video data as well as acoustic communication between these sites. Through MedNet, neurologists could monitor EEG signals from patients during surgery or intensive care and give immediate feedback to the personnel in the OR or at the bedside. In addition, several conference and lecture halls were connected to the MedNet to facilitate research and education. MedNet had a heterogeneous infrastructure including an Ethernet network for real-time transmission of neurophysiological and acoustic data and an analog NTSC video routing network which also contained an audio channel. The analog network only spanned three of the participating institutions.

MedNet implements some of the features which are required from an integrated OR: It covers the transmission of biosignals as well as audio and video streams through a heterogeneous infrastructure. MedNet can be seen as a proof of concept for the added value for the quality of clinical care but also for teaching which is gained from the integration of communication technology into the OR which builds bridges between the "islands of information". Nevertheless, the MedNet is a hardwired setup which connects a static set of rooms and devices. The MedNet does not show the flexibility regarding setup and dismantling of devices and definition of novel setups.

ORF – The Operating Room of the Future

The Operating Room of the Future (ORF) project at the CIMIT center at Massachusetts General Hospital in Boston is a highly ambitious project which aims at the construction of a prototypical solution of a fully functional integrated OR for minimally invasive surgery. The ORF development is based on the collaboration between academic researches and clinicians at CIMIT and several medical device companies. The ORF is meant to be a "*living laboratory*" in which concepts for system integration can be tested out and their impact on the surgical workflow can be investigated. Sandberg presented the state of development and a research agenda for the future enhancements of the ORF in 2003 [Sandberg *et al.*, 2003]. He summarizes that while they were able to reduce perioperative procedure times by up to 60% in abdominal and laparoscopic interventions, "the current state of technology is woefully unready for integration" ([Sandberg *et al.*, 2003], page 59). He points out that in order to integrate surgical devices it was necessary to bring traditionally competing vendors together in order to define interfaces. Sandberg does not give away any technical details about the tech-

nology which is used inside the ORF to link imaging devices, navigation devices, displays, control panels, and a teleconferencing system together. His remark on page 59 regarding the necessity for cooperation between vendors and his vision of a future version of the ORF in which devices are *"fully compatible at the software level"* while *"fully modular at the hardware level"* on page 62 may suggest the assumption that the integration is based on hand-tailored interfaces between specific devices rather than a unified infrastructure.

In the same year, Meyer et al. presented the integration of a central patient data display in the ORF with the patient databases in the HIS [Meyer et al., 2003]. They motivate a number of applications which could benefit from this integration, including an RFID-based system which ensures that the right patient is in the right room at the right time, an allergy warning system which accesses the patient's history before drug administration, and an integrated online ordering and reporting system. The approach the group around Meyer follows does allow access from the OR to the databases and infrastructures which are established in hospitals. It does not include the extension of the capabilities of these databases and infrastructures in order to enable the storage and transfer of data which is specific to surgery, such as surgical treatment plans.

OrthoMIT

The OrthoMIT project at RWTH Aachen aims at the development of an integrated platform for orthopedic interventions. The focus of the project lies on intraoperative imaging and navigation for hip-, knee-, and spinal surgery as well as trauma surgery. One aspect of the OrthoMIT project is the development of an integrated surgical workstation through which a surgeon has access to and control over intraoperative imaging devices, robotic tools, tracking systems, planning systems, and other CAS modules [Ibach et al., 2006]. As of 2006, no results in this regard were published by that group.

Commercial Integrated OR Suites

Several companies have acknowledged the surgical requirement for better integrated and more ergonomic OR environments. As of 2006, four vendors are offering integrated OR suites. Karl Storz' OR1, Olympus' EndoAlpha, Stryker's iSuite, and the Brainlab Brainsuite implement a multitude of functionalities, including video routing, PACS and HIS access, surgical navigation, intraoperative imaging, and a centralized input/output device for visualization and interaction in the sterile field. These systems have in common that they are not or only in some aspects based on open standards but are integrating devices using proprietary interfaces. This limits the flexibility to integrate additional devices into these suites once they are installed to products which are certified by the manufacturer of the suite. Such a limitation stands in contrast to the requirements for an open distributed infrastructure for the OR as they were postulated, e.g., at the OR 2020 workshop (see Chapter 1).

2.2.2 TIMMS

In [Lemke & Vannier, 2006] and [Lemke & Berliner, 2007], the Therapy Imaging and Model Management System (TIMMS) was proposed as a meta architecture for system integration in MGT. In 2007, TIMMS was accepted by the members of DICOM Working Group 6 (Base Standard) as *the* reference architecture for development of standards for the exchange of data between devices in the OR. TIMMS consists of three kinds of components: engines, repositories and a communication backbone. Figure 2.1 shows the elements and structure of the TIMMS meta architecture.

TIMMS Engines

In computer science, an engine is a part of an information processing system which acts independently. Engines are usually reusable components which are capable of performing operations of a certain type. Prominent examples are the physics engine in computer games or other software which perform all kinds of physical simulations that are required by other components of the software. Lemke et al. identify seven engines which provide the required capabilities of an MGS system and can be developed and operated independently [Lemke & Berliner, 2007]:

- Intraoperative imaging and biosensors: Devices that generate patient data from measurements. The imaging and biosensors engines provide access to the data sources in the OR.
- Modeling: Modeling engines create patient specific models from patient data and clinical knowledge.
- Simulation: Generic and patient specific models are used to predict the results of surgical decisions.
- Workflow management and decision management: Software which aims to the support of decisions by collecting, processing, and offering information about surgical workflows.
- Visualization/Representation manager: Display of information and models.
- Intervention: Functionalities which directly relate to surgical processes, such as the control of a telemanipulator or access to measured coordinates from a navigation device.
- Validation: Modules which validate the performance, correctness, and consistency of the models which are generated and processed by the other engines.

TIMMS Repositories

Repositories are integrated hardware and software components which store and manage the access to stored data and/or tools. There exist associated repositories that are

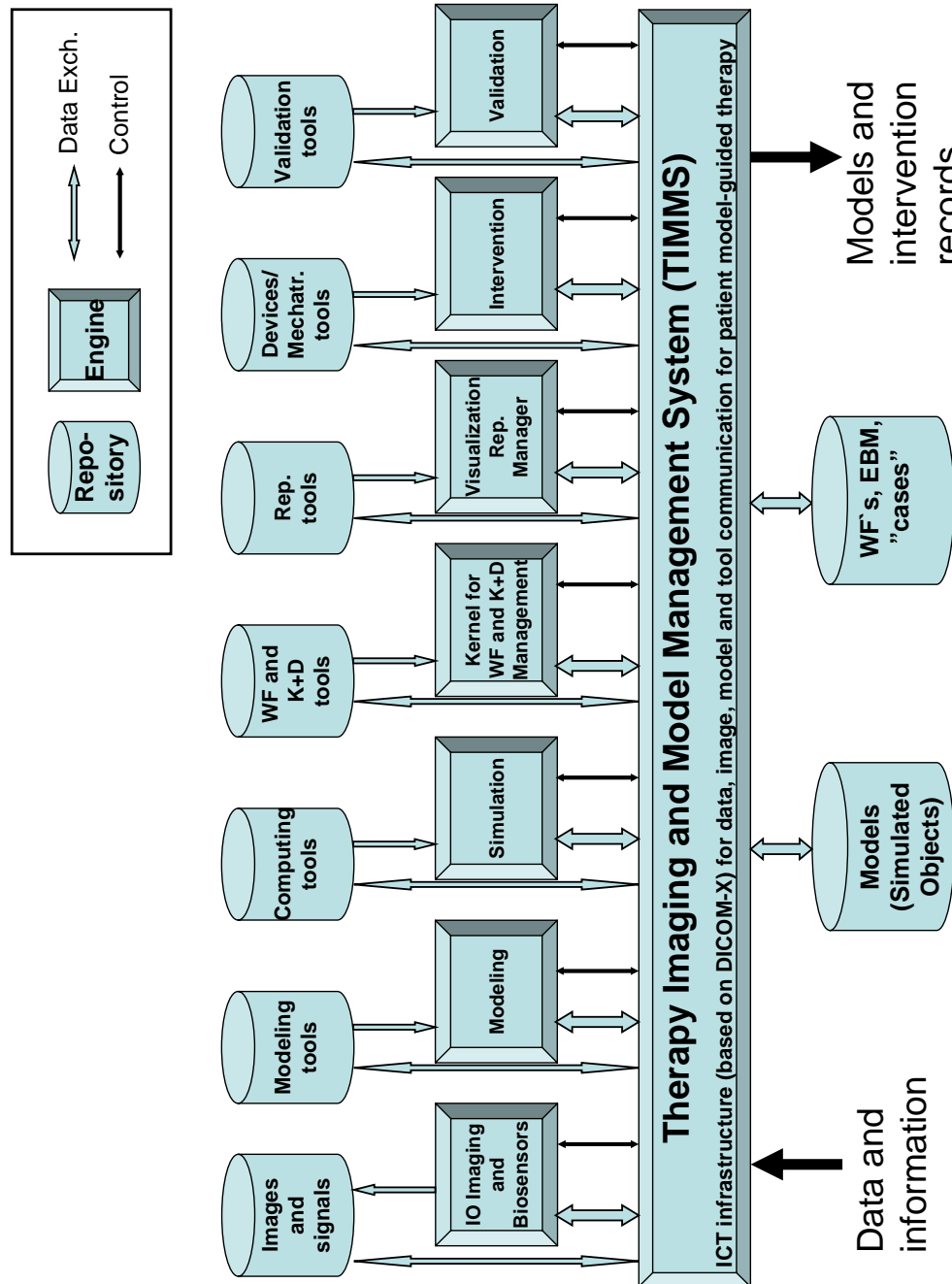


Figure 2.1: Structure of the Therapy Imaging and Model Management System (TIMMS) [Lemke & Vannier, 2006].

linked to the engines and unassociated repositories which are accessed from all engines via the TIMMS network backbone. The associated repositories may, in addition to their connection to an engine, be connected and accessible via the TIMMS backbone as well.

TIMMS Backbone

The TIMMS backbone is the communication infrastructure through which the engines retrieve data from repositories and communicate with each others. The communication over the backbone is performed with the means of standardized protocols, data encodings, and interfaces. The backbone has to implement transmission services compliant to a wide range of requirements. Real-time communication of small packages containing status information, measurements, or control sequences for mechatronic assistance systems need to be processed alongside with heavy data objects such as multi-slice 3D+t images. HD Video routing between image sources, processing and visualization engines requires real-time communication of MB-packages at a frame rate of 25 – 60 *Hz*. The bandwidth of requirements will in most scenarios be too wide to be optimally supported by one single technology. In a concrete architecture, the backbone may consist of several parallel infrastructures. This heterogeneity is to be transparent for the engines which exchange data through the backbone on the application level.

2.2.3 Standards and Protocols

The integration of information systems into an open infrastructure which enables the exchange of information between system from different manufacturers depends on the existence and pervasion of industry standards. Various standardization bodies have published regulations for interfaces and data structures in medical informatics. In the following, a list of standards is presented which might affect in the future or already do affect the integration of CAS systems.

DICOM

The DICOM (Digital Imaging and Communications in Medicine) standard has evolved to be the leading standard for storage, query and transfer of medical images since it started its evolution in the 1980s. Nowadays image acquisition devices, PACS servers, radiology workstations and reporting systems utilize data exchange based on the DICOM standard (see, e.g. [Horii & Bidgood, 1992; Yoo *et al.*, 1997; Huang, 2004; Dreyer *et al.*, 2005; dcm4chee, 2005]). As imaging technology evolves with time, the DICOM standard undergoes constant changes and extensions. In its 2008 version, DICOM specifies data structures and services for images of various modalities, segmented images and rigid as well as non-rigid spatial registration between image spaces. With DICOM Structured Reports (DICOM SR), results of diagnostic procedures can be represented in a structured, yet still human readable way. Recently, DICOM was extended to implement use cases from radio therapy [Germond & Haefliger, 2001].

DICOM worklist management facilitates scheduling of procedure steps in remote machines as well as notification of their execution.

The DICOM standard is maintained by the Medical Imaging & Technology Alliance (MITA), a division of the National Electrical Manufacturers Association (NEMA). After almost 20 years under the governance of these institutions, the International Standards Organization (ISO) made DICOM an ISO standard (ISO 12052:2006).

A brief introduction into the concepts of DICOM and DICOM nomenclature is given in Appendix B. For deeper insights, the reader is referred to the literature, e.g. [Pianykh, 2008], or the DICOM standard publication [NEMA, 2008b].

Surgical DICOM

In 2004, the DICOM Committee, which is the legislative power in the maintenance of DICOM, installed DICOM Working Group 24 (WG24) which aims at including surgical use cases into DICOM. WG24 began its work with an analysis of requirements for a surgical communication standard based on a collection of use cases from several surgical disciplines as well as interventional radiology (see Table 2.1). These use cases were presented in a white paper [H. U. Lemke (edt.), 2006] which serves WG24 as a fundament for the selection and specification of work items. The white paper contains detailed workflow descriptions of these surgical techniques. An example for a surgical workflow is presented in Figure 2.2 on the example of minimally invasive mitral valve reconstruction. In the left image, the whole procedure is visualized as a linear workflow on a very high level of abstraction. The right image shows one procedure step rolled out as a more precise workflow. There, the data which is generated or required by work steps is visualized as green parallelograms.

The white paper identifies existing DICOM SOP classes for these data objects, which are appropriate for implementing communication and archival for those data objects where such a SOP class could be identified. Several of the data objects for which no SOP class could be identified could be referred to work items which were under conduction by other DICOM working groups at the time the white paper was written. For example, several chapters in the white paper refer to the segmentation IOD which was not part of the standard until 2006. Finally, the white paper motivates a number of potential new work items. Among these, data structures for polygonal surfaces and intraoperative video exchange are addressed most often.

Health Level 7

Health Level 7 (HL7) is a standard that provides interconnectivity of Hospital Information Systems (HIS), Laboratory Information Management Systems (LIMS), Electronic Billing Systems (EBS) and Electronic Health Records (EHR). The most pervaded version of HL 7 in hospital information systems of the present is version 2 which was released in 1987. HL 7 2.x (2.1 – 2.6) specifies a protocol for message exchange between applications in health care facilities. HL 7 uses text-based encodings of content which is described with the means of standardized key-words, codes and encodings.

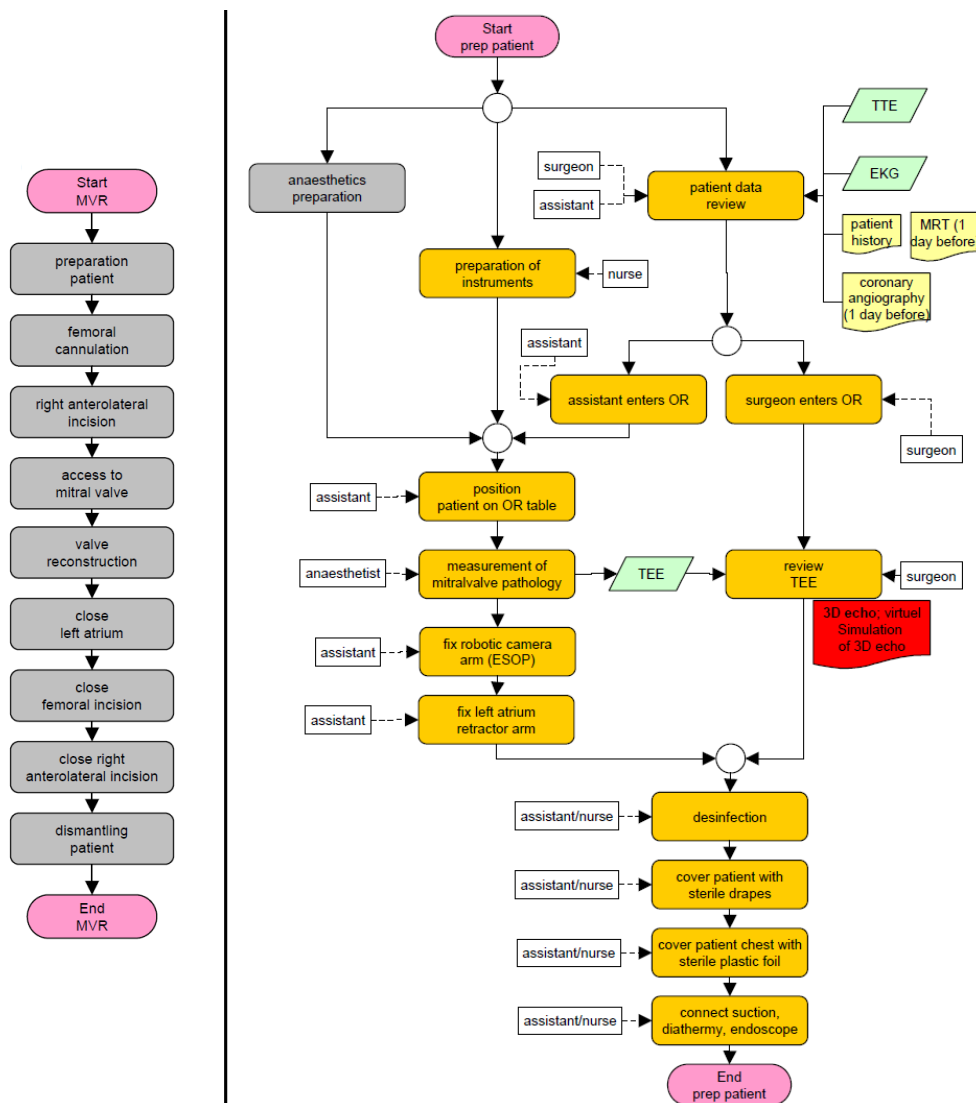


Figure 2.2: Top level workflow (left) and detailed workflow of the first work step of mitral valve reconstruction (right) [H. U. Lemke (edt.), 2006].

Version 3 of HL 7 was released in 2006. It specifies an object-oriented Reference Information Model (RIM) which explicitly models the data structures required in HIS and the relations between these data structures. The RIM is intended to facilitate the interpretation of messages and reduce development effort for HIS manufacturers. Message exchange in version 3 is based on the Clinical Document Architecture (CDA), an XML-based markup standard for the encoding of messages. The pervasion of HL 7 version 3 is only evolving very slowly, mostly due to the incompatibility with version 2 and the fact that most people are relatively satisfied by the functionalities version 2 provides.

Discipline	Intervention
Cardiac Surgery	Minimally Invasive Mitral Valve Reconstruction
	Total Endoscopic Coronary Artery Bypass Grafting
Neurosurgery	Craniotomy with Navigation Support
ENT Surgery	Micro Laryngeal Surgery
	Foreign Body Excision
Orthopedic Surgery	Total Hip Replacement
Thoraco-Abdominal Surgery	Laparoscopic Splenectomy
	Laparoscopic Cholecystectomy
	Laparoscopic Nephrectomy Left
Interventional Radiology	Peripheral Angiography with Intervention
	CT Guided Hepatic Tumor Radio-Frequency Ablation
	Transjugular Intrahepatic Portosystemic Shunt

Table 2.1: Surgical workflows in WG24 white paper [H. U. Lemke (ed.), 2006].

VITAL / ISO 11073

The Health Informatics - Vital Signs Information Representation - VITAL standard was released by the European Committee for Standardization (CEN) and aims at the integration of biosignal monitoring devices for data collection in intensive care units and for telemedicine applications. VITAL specifies a Domain Information Model (DIM) for the description of biosignals, an communication model based on the ISO/OSI layer model and a standardized nomenclature for medical terms [ENV, 2000].

VITAL was adopted by the ISO standard "Health informatics – Point-of-care medical device communication" (ISO 11073) [ISO, 2006]. With the exception of some prototype implementations (e.g. [Anagnostaki *et al.*, 2001]), VITAL and ISO 11073 never really were pervaded in practical applications. The major critique which was adduced as a reason for this was the impression that both are extremely complex to implement and far too complicated for the applications of most vendors.

Nevertheless, the information model and standardized nomenclature which is contained in ISO 11073 should be considered when standardizing the semantic and semi-otic background for the exchange of biosignals in the OR.

IHE

Integrating the Healthcare Enterprise [IHE, 2009] is an initiative by healthcare providers and system manufacturers which aims at increasing the interoperability between devices. IHE is not a standardization body. IHE *Integration Profiles* are technical documents which propose good-practice workflows and architectures for

the data exchange in different domains of healthcare in a not normative manner. IHE makes recommendations on how existing standards should be used in a way which optimally facilitates interoperability.

IHE is organized in technical committees which focus on one domain of application. At the time of writing this thesis, no domain which covers the integration of perioperative and intraoperative data exchange is registered.

Chapter 3

Dataflow in CAS

In this chapter, the requirements for the integration of surgical planning software and CAS modules into an information system which is based on non-proprietary interfaces are investigated. Sandberg et al. [Sandberg *et al.*, 2003], Burgert et al. [Burgert *et al.*, 2007], and DICOM Working Group 24's white paper [H. U. Lemke (ed.), 2006] recommended an approach where these requirements are derived from surgical workflows. Neumuth et al. have proposed and applied a method where surgical workflows are created on the basis of actually observed surgery protocols [Neumuth *et al.*, 2005, 2009]. Some of the workflows presented in the white paper are based on their findings. Based on this work and the works of Strauß, Meixensberger, Jacobs, Blecha, et al. [Strauß *et al.*, 2008; Meixensberger, 2008; Jacobs *et al.*, 2008; Blecha *et al.*, 2007], and additional input from clinical experts, the author analyzed the data flow requirements for the following surgical workflows:

- Total Hip Replacement (THR)
- Mitral Valve Reconstruction (MVR)
- Transapical Aortic Valve Implantation (TA-AVI)
- Navigated Control Functional Endoscopic Sinus Surgery (NC-FESS)
- Brain Tumor Surgery (BTS)

3.1 From Workflows to Dataflows

The derivation of dataflow requirements from surgical workflow was performed in three steps. Firstly, the information regarding data exchange which was already contained in some of the workflows in an incomplete manner was amended in cooperation with expert surgeons and with technical experts from the industry. An amended version of the workflow from Figure 2.2 which contains the input and output data for each work step is shown in Figure 3.1.

Secondly, the presentation of the acquired information is transformed from a process-centric workflow model to data-centric Data Flow Diagrams (DFD, see Appendix A).

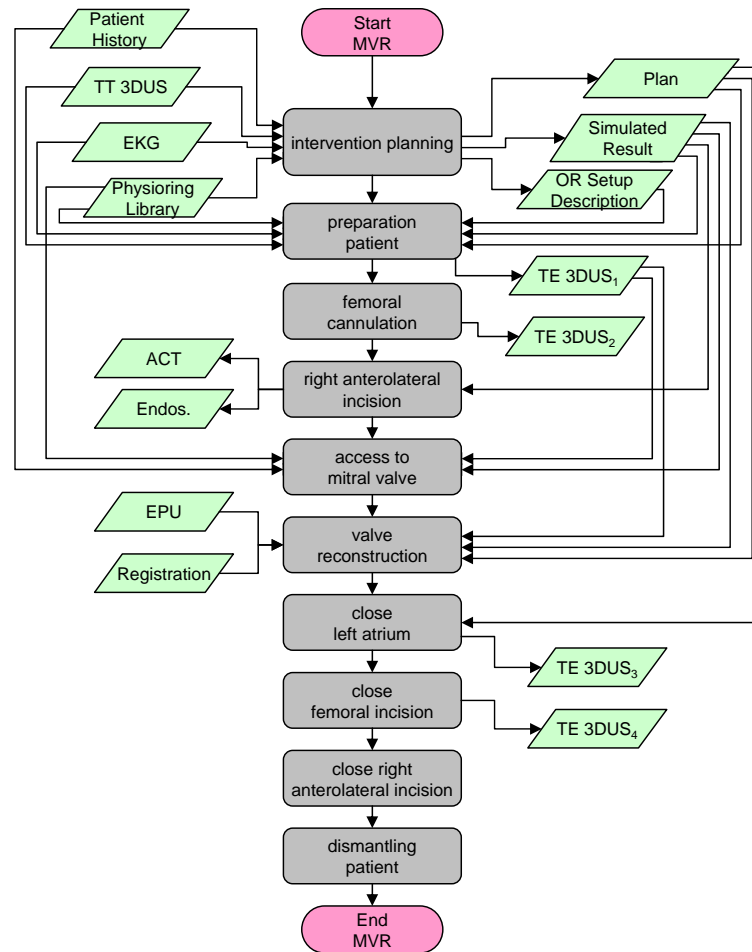


Figure 3.1: Workflow and dataflow for mitral valve reconstruction.

Thereby, the data flow was classified according to a five-phases model. Surgical procedures are commonly described in three phases (e.g. [Sandberg *et al.*, 2003]):

- The preoperative phase during which diagnosis and intervention planning are performed.
- The intraoperative phase during which the intervention is conducted.
- The postoperative phase during which the OR report is written, the surgical outcome is assessed, and the patient is monitored until dismissal.

The information systems, imaging devices, and processing workstations which are employed during these phases are, in general, disparate. Nevertheless, there is a requirement for the information created in one phase to be transferred into the subsequent phase. For the analysis of the dataflow during and between the three perioperative phases, the data exchange can be classified according to five phases of dataflow:

- Phase A: Preoperative images and data are transferred to the diagnostic and planning workstations.
- Phase B: Preoperatively acquired data and treatment plans are transferred into the OR.
- Phase C: Intraoperative data exchange inside the OR.
- Phase D: Intraoperatively acquired or generated data is transferred from the OR to systems outside of the OR.
- Phase E: Postoperative assessment and patient monitoring.

A very generic data flow diagram (DFD) for CAS is presented in Figure 3.2 on the left. It represents the "zero-level" of granularity which is abstract enough to apply to all the workflows which were considered. The diagram contains the exchange of data during phases A, B, D, and E. The diagram reflects current situation as of 2006 in a not integrated perioperative CAS scenario. The data exchange between preoperative planning, intraoperative assistance, and postoperative follow-up is mostly based on hardcopies or proprietary files on mobile storage media. On the right hand side of Figure 3.2, the data flow during Phase C of computer assisted TA-AVI is presented. The cDFDs shows to which extent the repositories and communication infrastructures available in the HIS or RIS are used by these procedures. The solid arrows in the DFD

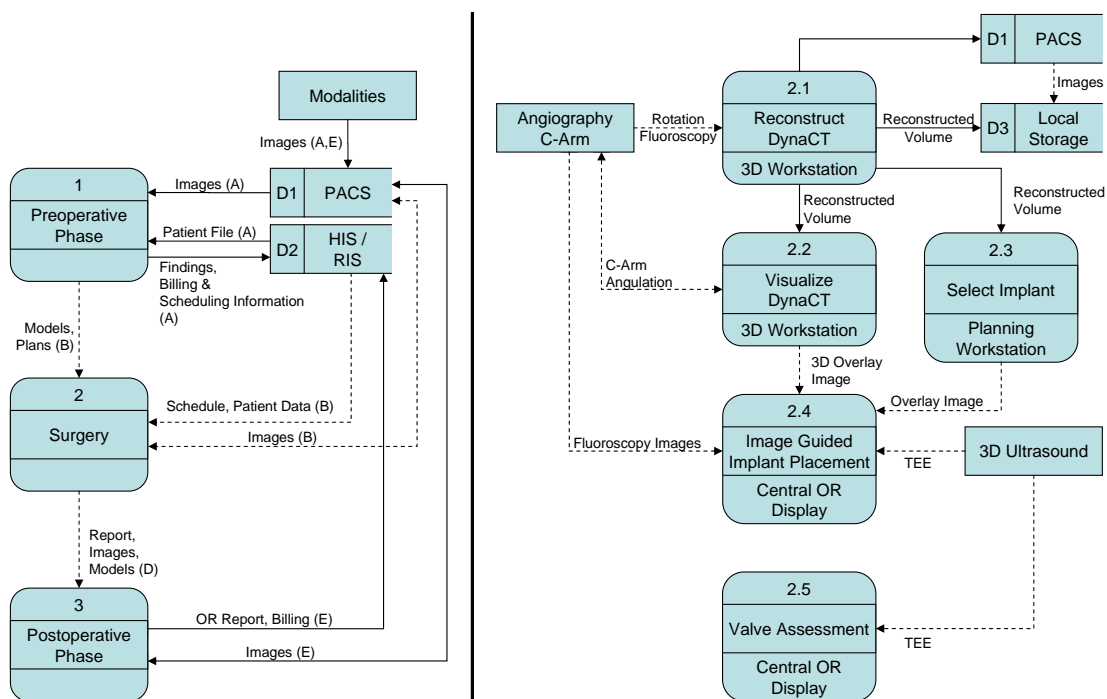


Figure 3.2: Left: Current dataflow between pre-, intra-, and postoperative systems. Right: Dataflow between processes during TA-AVI.

represent data flow for which standardized interfaces are available and implemented in several commercially available solutions. The dotted arrows represent data flow which is implemented via paper files or digital data exchange on the basis of proprietary interfaces.

Thirdly, a list was generated on the basis of the DFDs which contains the required information exchange for all five workflows. The focus of this step was limited to phases B, C, and D. For phase A, the existing standards and interfaces in the HIS, RIS, PACS and other clinical information systems are sufficiently covering the requirements. Surgical planning workstation need to implement these standards in order to access the information. After outputting the data which was acquired or generated to the HIS and its subsystems in phase D, the existing information systems for intensive care units, patient billing, and follow-up diagnostics are applicable in phase E. It is a requirement, that the data which is output during phase D is encoded in a way which facilitates further processing by the information systems which are employed postoperatively.

In order to identify standardized means of data exchange which could be utilized to implement the data flow during phases B, C, and D, the properties of the data which is exchanged and the requirements to the transfer were investigated. A table was created which classifies the data flows according to the following criteria:

- The type of the data. One of seven types was assigned: Image, geometry, biosignals, structured document, text document, status, other.
- The phase(s) during which the data flow occurs.
- Data is either patient-specific or not patient specific. Patient-specific data contains a reference to a particular patient, e.g. a patient ID or patient name.
- Periodically exchanged data, e.g. biosignals which are sampled in regular intervals or video signals which are transmitted at a certain frame rate, is classified as *stream*. Data flow which occurs singularly or multiply in arbitrary intervals, such as user input or event notifications, is classified as *message*.
- Object size: the average or typical size of one data element (applies to messages).
- Bandwidth: the average or typical bandwidth which is required by a stream (applies to streams),

3.2 Data Exchange Requirements

In Table 3.1 the list of required data flows for the five investigated surgical workflows is presented.

Name	Occurs in	Type	Phase(s)	Patient Specific	Average Size / Bandwidth	Stream or Message
Implantation Plan	THR, TA-AVI, MVR	Text document, structured document, or other	B	Yes	1 MB – 10MB	Message
Incision Planning	TA-AVI, BTS	Geometry	B	Yes	1 MB – 10 MB	Message
OR Setup Description	THR, MVR, TA-AVI, BTS, NC-FESS	Text document, structured document, or other	B	Yes	1 MB – 10MB	Message
Aortic Annulus Radius	TA-AVI	Text document or structured document	B	Yes	< 1 MB	Message
Electrophysiology Report	MVR	Text document or structured document	B	Yes	< 10 MB	Message
C-arm Angulation	TA-AVI	Status	C	No	< 1 MB/s	Stream
C-arm Control	TA-AVI	Status	C	No	< 1 MB/s	Stream
Foot Pedal Signal	NC-FESS	Status	C	No	< 1 MB/s	Stream
Navigation Coordinates	THR, NC-FESS, BTS	Status	C	No	< 1 MB/s	Stream
Power Source Control	NC-FESS	Status	C	No	< 1 MB/s	Stream
Registration	THR, MVR, TA-AVI, BTS, NC-FESS	Status	C, D	Yes	< 1 MB	Message
Arterial Clotting Time	MVR	Biosignal	C	Yes	< 1 MB	Message

Table 3.1: Data Types for THR, MVR, TA-AVI, NC-FESS, and BTS. (Continued on next page)

Name	Occurs in	Type	Phase(s)	Patient Specific	Average Size / Bandwidth	Stream or Message
ECG	THR, MVR, TA-AVI, BTS, NC-FESS	Biosignal	C, D	Yes	< 1 MB/s	Stream
Digital X-ray	THR	Image	B	Yes	< 10 MB	Message
CT	THR, TA-AVI, MVR, NC-FESS	Image	B	Yes	< 1GB	Message
MRI	THR, MVR, TA-AVI, BTS	Image	B	Yes	< 1 GB	Message
Live X-ray fluoroscopy / angiography	THR, MVR, TA-AVI	Image	B	Yes	10 – 20 MB/s (uncompressed)	Stream
Recorded X-ray fluoroscopy / angiography	THR, MVR, TA-AVI	Image	C	Yes	< 1 GB	Message
3D X-ray	THR, TA-AVI	Image	B	Yes	< 10 MB	Message
Live 2D Ultrasound	BTS	Image	C	Yes	10 – 20 MB/s (uncompressed)	Stream
Reconstructed 3D(++) Ultrasound	BTS	Image	C,D	Yes	< 1 GB	Message
Recorded 3D+t Ultrasound	MVR, TA-AVI	Image	B,C,D	Yes	< 1 GB	Message
Recorded 2D+t Ultrasound	MVR, TA-AVI	Image	B,C,D	Yes	< 1 GB	Message
Endoscopy	MVR, NC-FESS	Image	C	Yes	PAL: < 20 MB/s (uncompressed)	Stream

Table 3.1: Data Types for THR, MVR, TA-AVI, NC-FESS, and BTS. (Continued on next page)

Name	Occurs in	Type	Phase(s)	Patient Specific	Average Size / Bandwidth	Stream or Message
Recorded Endoscopy Video	MVR, NC-FESS	Image	D	Yes	Up to Giga-bytes	Message
Endoscopy Snapshots	MVR, NC-FESS	Image	D	Yes	< 10 MB	Message
Overlay Images for AR	TA-AVI, BTS	Image	C	Yes	< 10 MB / s	Stream
2D Implant Templates	THR	Geometry	B	No	< 10 MB	Message
3D Implant Templates	THR, MVR, TA-AVI	Geometry	B	No	< 10 MB	Message
2D Landmarks	THR	Geometry	B	No	< 10 MB	Message
3D Landmarks	THR, TA-AVI, MVR, NC-FESS	Geometry	B	No	< 10 MB	Message
Surface Models of Anatomy	THR, TA-AVI, NC-FESS, BTS	Geometry	B,C,D	Yes	< 100 MB	Message
Workspace	NC-FESS	Geometry	B	Yes	1 MB – 10 MB	Message

Table 3.1: Data Types for THR, MVR, TA-AVI, NC-FESS, and BTS.

Phase B, the transfer of preoperative images, models, and plans, is dominated by the exchange of documents, images, or other datasets between workstations, repositories and intraoperative systems. The requirements to the communication infrastructure during this phase are very similar to the requirements during phases A and E, during which HIS, RIS, PACS, and other information systems are utilized: The importance of data integrity and durability is superior to performance requirements. Interoperability is important, since surgical planning software from different vendors needs to be integrated. Changes of clinical practice or investments in new technologies will require the information system to be flexible enough to allow extensions. The incidence of integration of new systems is not very high. Therefore, the availability of plug-and-play functionalities is a minor requirement. In Chapter 4, new DICOM data structures are presented which extend the domain of the DICOM standard towards storage and transfer of surgical planning data.

Phase C, the intraoperative phase, is dominated by soft- and hardware systems acting and communicating simultaneously under real-time conditions. The requirements during this phase are similar to an automation system or a video conferencing system: Data integrity is as important as in-time delivery of data. Streaming communication relies on guaranteed frame-rates. Intraoperative information systems are usually ad-hoc systems, i.e. they are set up immediately before used and dismantled afterwards. One of the key requirements to an OR infrastructure is auto-configuration of devices, or a "surgical plug-and-play" mechanism [Cleary & Kinsella, 2004]. Inside the OR, no long-term archives are located. Nevertheless, situations can occur, where access to external repositories is required in order to retrieve patient data which was not imported during Phase B. A software library is presented in Chapter 5 acts as an application level interface for service discovery and peer-to-peer communication inside the OR.

Chapter 4

Surgical DICOM

In order to eliminate the media disruption between preoperative diagnostics and planning and the intraoperative display and implementation of surgical plans, it is a requirement to integrate the OR into the information systems which are utilized during planning and diagnosis. For image based intervention planning, this means to build a bridge between the OR and the PACS. Inside the PACS, images, image-related information, and commands are usually communicated with the means of the DICOM standard. DICOM originates in diagnostic radiology. Its original intent was to act as a vendor independent file format and network interface for storage and exchange of radiology images and streamlining of radiology workflows. Its scope has been extended to storage and transfer of treatment plans in the field of radiotherapy in the 1990s [NEMA, 1996]. Several publications discuss the implementation of this standard in real-world systems and its effect on the radiotherapeutic workflow [Germond & Haefliger, 2001; Law & Huang, 2003; Law & Liu, 2009; Law *et al.*, 2009]. In 2005, DICOM Working Group 24 (Surgery) was founded with the aim to extend the DICOM standard with respect to use cases from computer assisted surgery. It is still a matter of discussion inside WG 24, to which extent the DICOM standard can provide the basis for the integration of devices in the OR (Phase C). In this chapter the focus lies on the application of DICOM to implement the data exchange during Phases B and D, i.e. the transfer of planning data into the OR and the export of intraoperatively recorded data to external repositories for long-term archival. In this chapter, two work items for the extension of DICOM are identified and the information models which were the base for their conduction are presented. These considerations are largely based on the data flow which was described in Chapter 3 for five concrete surgical interventions. This practice is in good agreement with the recommendation from the DICOM Working Group 24 White Paper [H. U. Lemke (edt.), 2006; Lemke, 2007], where an approach is motivated which is based on existing, good-practice workflows in order to guarantee for all standardization efforts to solve real-world problems in a fashion which matches realistic clinical requirements.

4.1 Identification of DICOM Work Items

The DICOM standard in its 2005 version already contains SOP classes for a large share of the data flow during Phase B as it was presented in Chapter 3. In Table 4.1, these data flows are listed. For those objects where this is possible, a DICOM IOD is named in the table which could represent the object. The remaining data flows were considered candidate work items for the extension of DICOM for transfer of preoperative data into the OR. These data flows are:

Name	IOD(s)
Implantation Plan	-
Incision Plan	-
OR Setup Description	-
Aortic Annulus Radius	-
Electrophysiology Report	- (partially covered by Basic Cardiac Electrophysiology)
Registration	Spatial Registration
ECG	12-Lead / General Electrocardiogram
Digital X-ray	Computed Radiography Image
CT	(Enhanced) CT Image
MRI	(Enhanced) MR Image
Recorded X-ray fluoroscopy / angiography	(Enhanced) XA/XRF Image
3D X-ray	Enhanced XA/XRF Image
Reconstructed 3D Ultrasound	Enhanced US Volume
Recorded 2D+t (Doppler) Ultrasound	Ultrasound Image
Recorded 3D+t Ultrasound	Enhanced US Volume
Recorded Endoscopy Video	Visible Light Image
Endoscopy Snapshots	Visible Light Image
2D Implant Templates	-
3D Implant Templates	-
2D Landmarks	-
3D Landmarks	-
Surface Models of Anatomy	-
Workspace	-

Table 4.1: DICOM SOP classes applicable for THR, MVR, T-AVI, NC-FESS. and BTS (based on [Löpfe *et al.*, 2006]).

- **Implantation Plan:** A patient-specific selection of implant templates, their registration to patient space as defined by preoperative images or models.
- **Incision Plan:** A patient-specific description, usually in geometric form, of the intended body opening through which an intervention is performed. The incision is determined according to preoperative images or surface models.
- **OR Setup Description:** A textual or structured document which lists the required equipment and equipment settings for an intervention.
- **Aortic Annulus Diameter:** Diameter of the aortic root, preoperatively measured on patient images. This value is required during TA-AVI to select the optimal diameters of balloon catheters and aortic valve implants.
- **Electrophysiology Report:** The MVR workflow contains an option where atrial arrhythmia is treated in addition to the reconstruction or replacement of the mitral valve. For patients who underwent electrophysiological mapping, the final mapping and treatment report from electrophysiology is presented during surgery.
- **2D and 3D Implant Templates:** Graphical representations of surgical implants which are utilized during preoperative planning to select the optimal implant and its position with respect to anatomical landmarks identified in patient images or surface models.
- **Surface Models of Anatomy:** Polygonal surface representations of organs or other structures, usually extracted from patient images.
- **Workspace:** The geometric definition of the space in which the shaver is intended to operate during NC-FESS.

Of these data types, surface models of anatomy, 3D implant templates, the NC-FESS workspace, certain aspects of the electrophysiology report, and the incision plan require a mechanism to describe geometric objects in patient space with the means of polygonal meshes. This is a very generic requirement for which DICOM does not contain a suitable data structure. The definition of a generic module for polygonal surface meshes in DICOM was identified as the foremost work item to be pursued by Working Group 24. Together with DICOM Working Group 17 (namely Scott L. Johnson from Philips Radiation Oncology Systems in Madison, Wisconsin, USA), the author of the thesis was engaged with the task to specify a DICOM SOP class for storage and transfer of surface segmentations. In Section 4.2, the resulting SOP class which was officially added to the DICOM standard as Supplement 132 [NEMA, 2008a] in 2008 is described.

Implantation planning was selected by WG 24 to be the second most pressing work item from that list. This includes data structures and services to describe 2D and 3D templates as well as patient-specific implantation plans. This work item was assigned to Working Group 24 in 2007. It was split into two work packages, of which the

definition of implant templates is pursued under supplement number 131 [NEMA, 2009a] by the author. The second work package regards storage and transfer of patient-specific implantation plans and is pursued by Thomas Treichel at ICCAS in Leipzig under supplement number 134 [NEMA, 2009b]. As of autumn 2009, both supplements are close to being released into the final voting process during which the decision is made about adding the proposed services and data types to the standard. The IODs which are proposed in supplement 131 are presented in Section 4.3.

The remaining data objects fall into two classes: The OR setup description is a highly technical document which clearly falls out of the domain of DICOM. In Chapter 5, an infrastructure for the integration of OR systems is presented. According to the recommendations of the OR2020 workshop and preceding meetings (see discussion in Chapter 1), one of the features of this infrastructure is a plug-and-play mechanism through which the components of a modular CAS system automatically identify their peer components in the network. In this scope, the issue of setup descriptions or integration profiles will be discussed. The aortic annulus diameter and the non-graphical aspects of the electrophysiology report are diagnostic findings which are based on image information. Within DICOM, the Structured Reporting (SR) mechanism is available for transfer and archival of this kind of data. SR documents are very generically defined and can potentially be used to encode any kind of clinical finding. The strength of DICOM SR is the possibility to exchange image-related diagnostic findings through the same infrastructure through which the images are exchanged rather than to require an additional EHR infrastructure, such as it is specified by OpenEHR or other standards. The weakness of DICOM SR is its relatively low pervasion, especially outside of radiology information systems. It is an open question which of the existing approaches should be preferably utilized to transfer diagnostic data into the OR. This decision can only be arrived by a committee or other body including representatives of the relevant industries as well as experts with a background in hospital management, hospital information systems, and medicine. Potentially, the IHE (see Section 2.2.3) initiative could be the foundation for such a panel. Within the thesis, no further regard is spent on this topic.

4.2 Surface segmentation SOP Class

The examination of data flows in Chapter 3 and Section 4.1 revealed seven data objects for which no DICOM SOP class is applicable. Five of these data objects include polygonal surface models. Another indicator for the importance of geometric modeling is their pervasion in CAS publications. During the Visualization, Image-Guided Procedures and Display session at the SPIE Medical Imaging Conference in 2006, for instance, 91 papers were presented of which 45 contained surface meshes [Gessat *et al.*, 2007].

DICOM is originally designed according to a world model which is founded in radiology workflows. In Appendix B, the DICOM model of the real world is depicted as an Entity-Relationship diagram in Figure B.3. This model contains an image Infor-

mation Entity (IE) which is an abstraction of all modality specific images covered by DICOM. In order to represent patient specific surface models, a surface IE is required on the same level as the image IE (see Figure 4.1).

While computer scientists favour to think in abstract categories or classes, the evolution

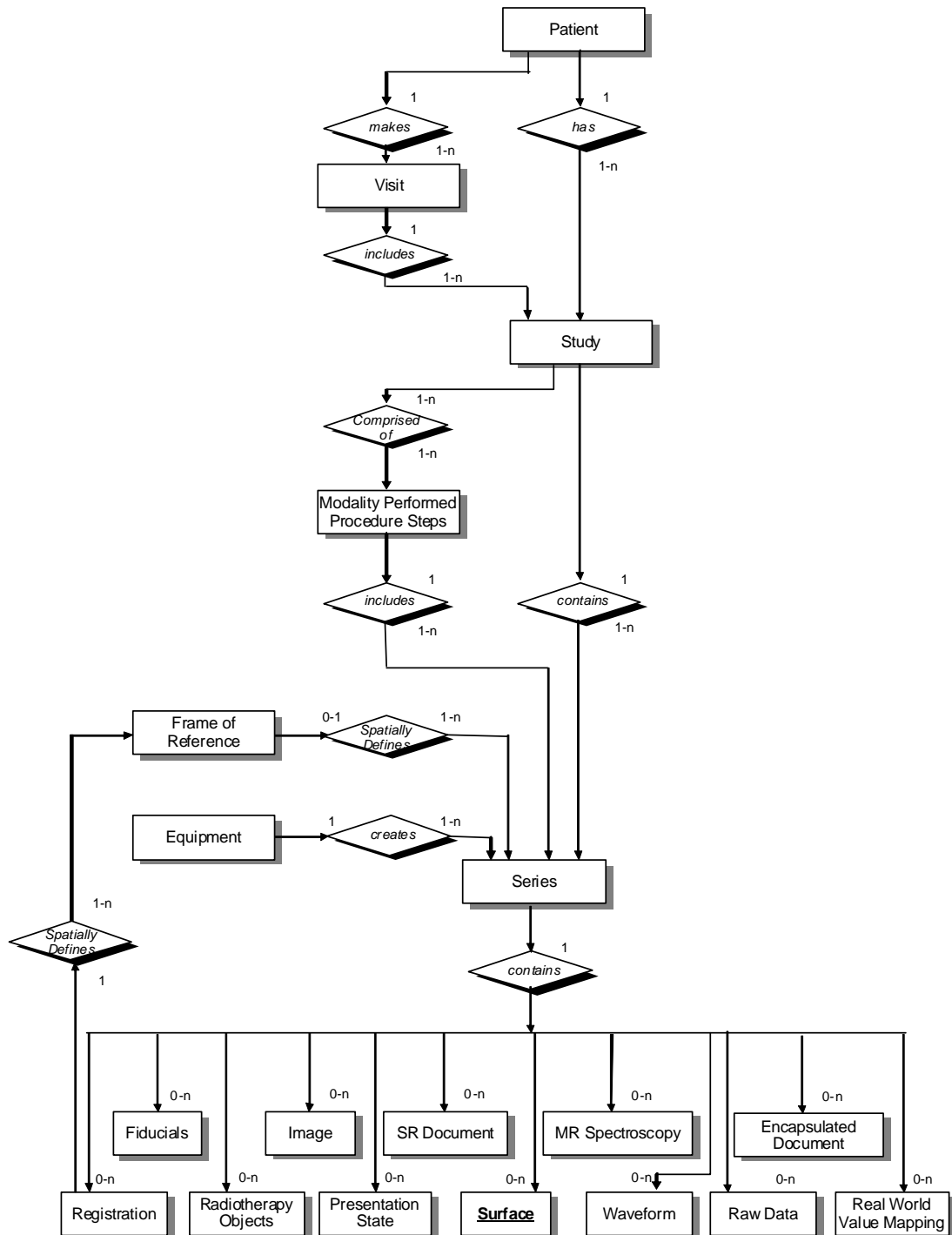


Figure 4.1: DICOM model of the real world including surfaces.

of the DICOM standard is strictly bound to a "one use case at a time" approach. The observation that there are several use cases which require polygonal surface models may be a good argument to start a work item on either one of these and to pursue it in a manner that allows re-using the geometric data structures in future use cases. Nevertheless, adding a required IE and the according modules to the standard would not be accepted by the DICOM Committee as a work item rationale. Instead, the work item rationale has to be a clinical use case with real-world applications in at least one major vendor's commercial products. For surface models, the most prominent use case in CAS is the storage and transfer of surface models which were derived from patient images in a segmentation process. Surface segmentation was selected as the "vehicle" for introducing the surface IE and, accordingly, a surface module to DICOM.

4.2.1 Requirements

The surface segmentation SOP class was specified to satisfy the following requirements:

- The geometric description of a surface segmentation shall be contained in a separable module that can be reused in other IODs which contain a surface description. The module has to contain mechanisms for the description of piecewise linear surfaces consisting of triangles or planar polygons ("*facets*"). Lines, piecewise linear curves ("*polylines*") and 0-dimensional nodes also have to be contained in the module.
- In most cases where surface meshes are used in clinical software, they are visualized to the user of a system. In order to visualize a surface mesh, the rendering engine requires material parameters, such as color and opacity of the surface mesh. While DICOM usually treats the exchange of visualization parameters separately from the representation of data, it was decided that for surfaces at least a minimal set of rendering parameters should be part of the data IOD.
- Segmentation, surface reconstruction, and diagnosis are in some settings performed on disparate systems by different users. This necessitates that surface segmentation instances contain information describing the context of the segments along with a description of how they were derived.
- Distances between structures, volumes of structures, and other geometric parameters are calculated for various purposes. Some of these calculations require knowledge about topographic or geometric properties of the surfaces, e.g. the information whether the surface is "waterproof", i.e. envelopes a finite volume. It is required that such parameters can be stored with a surface mesh.

4.2.2 Surface Segmentation Class Diagram

The class diagram in Figure 4.2 shows a model of the information which is contained in the surface segmentation IOD as a UML class diagram. It contains the following classes:

- The Patient, Study, Series and Equipment classes represent the DICOM modules which reflect the DICOM information model for patient data.
- The FrameOfReference class assigns a globally unique identifier (a UID in DICOM context) to a spatial Frame Of Reference (FOR). When two DCOM instances share the same FOR, the same spatial coordinate in both instances correspond to the same location in the patient. The rigid or non-rigid registration IODs can be used to express the spatial relations between different FORs.
- The Segment class describes one segment of the patient space. A segment represents any kind of property which can be, e.g., an organ, an area of high activity in a functional image, or a contrast agent bolus. The Segment class contains a description of this property using standardized codes and in addition free-text descriptions. A segment references one or more surfaces, i.e. a segment can be an aggregation of more than one geometric item.
- The SegmentSurface class establishes the reference between segments and

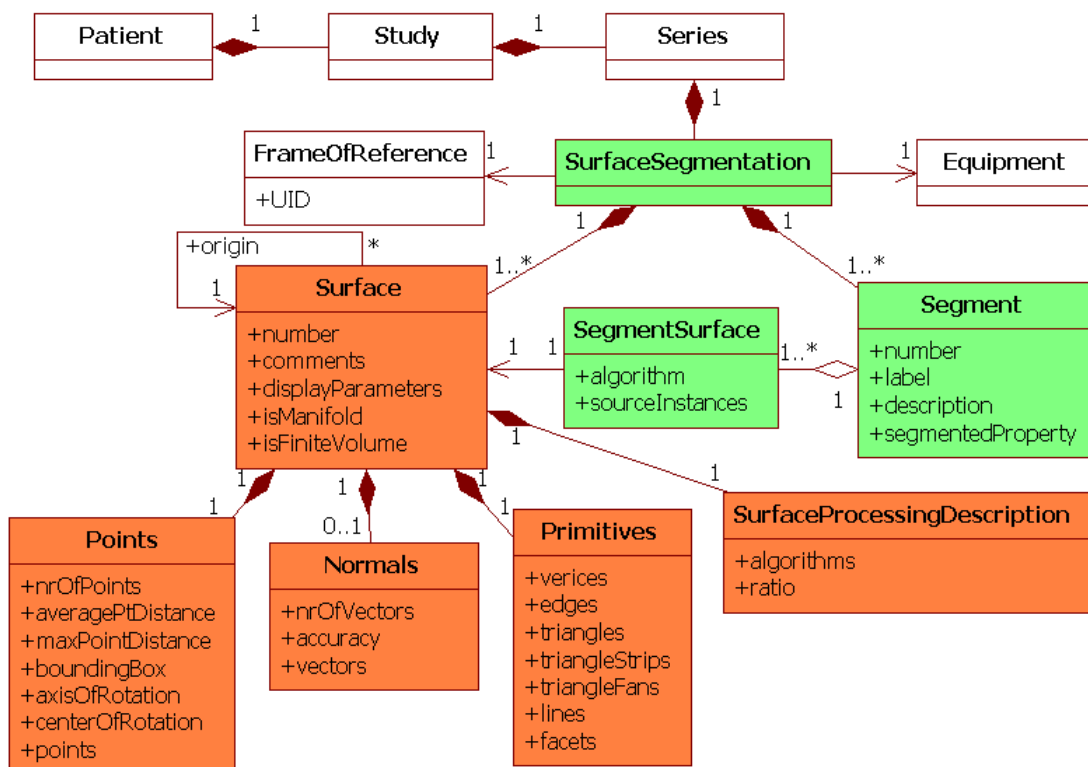


Figure 4.2: UML class diagram of the surface segmentation.

the surfaces used to construct the segments. Each `SegmentSurface` contains a list of the DICOM instance references and descriptions of the algorithms which were used to generate the surface.

- The `Surface` class describes the geometric properties of a surface mesh. Its points P are contained in a `Points` object, its primitives T are contained in a `Primitives` object. If present, an object of the `Normals` class adds a vector to each point in P which indicates the surface normal in that point.
- The `SurfaceProcessingDescription` class describes the derivation of a surface based on other surfaces. A description of the derivation algorithms is contained in standardized codes. There is also a description of the degree of processing and a reference to the original source.

4.2.3 Surface Segmentation Storage SOP Class

DICOM supplement 132 specifies the surface segmentation storage SOP Class, consisting of a surface segmentation IOD and services to store, transfer and query instances of that IOD. The modules which are contained in this IOD are listed in Table E.1. Supplement 132 specifies two new modules which contain the actual segmentation information and geometric information. A surface segmentation IOD instance is a serialization of an object of the `SurfaceSegmentation` class and associated objects according to the UML class diagram in Figure 4.2.

- The surface mesh module geometrically and topologically describes one or more polygonal surfaces. A complete list of the attributes in this module is presented in Table E.3. It is represented by the orange classes in Figure 4.2.
The surface mesh module is specified as a general module which is intended to be present in IODs other than the surface segmentation IOD. For that reason, the module does not contain any attributes which are specific to segmentation.
- The surface segmentation module adds semantics to the geometric entities specified in the surface module. A complete list of the attributes in this module is presented in Table E.2. It module is represented by the green classes in Figure 4.2.
Included semantics are: properties represented by the surface, references to image data, and algorithms used to derive the surface from the image data.

To guarantee that the description of the segment semantics is commonly understandable, DICOM provides standardized codes to identify properties as well as algorithms. Where applicable, SNOMED CT codes are utilized. Where no appropriate SNOMED CT codes exist, DICOM codes are specified in supplement 132. Figure 4.3 shows an example of how surfaces and segments are stored in the IOD. In 2008 the development of DICOM supplement 132 was finished. The surface segmentation SOP class was officially published as a part of the DICOM standard in its 2010 version.


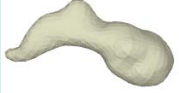

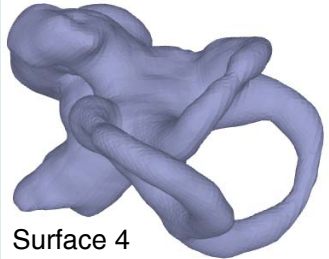
Patient Module			
Name: John Doe ID: 001		born: January 1st, 1950 sex: male	
General Study Module			
Study Instance UID: 1.1.1.1.1.1 Study Date: 06.08.2008		Study ID: 5 Accession Number: 1	
General Series Module			
Modality: SEG Series Instance UID: 1.1.1.1.1.2		Series Number: 3 Laterality: R	
Frame of Reference Module			
Frame of Reference UID: 1.1.1.1.1.3			
Surface Mesh Module			
Surface 1 	Surface 2 	Surface 3 	 Surface 4
Surface Segmentation Module			
Segment Nr.	Surfaces	Label	Segmentation Type
1	1,2,3	Ossicular Bones	Manual
2	4	Ductus semicirculares	Manual

Figure 4.3: Surface segmentation IOD instance.

4.3 Implant Template SOP Classes

In implantology, the preoperative selection of the best fitting implant and its virtual placement in patient space is called implantation planning. Thereby, 2D or 3D radiology images of the patient are visualized together with graphical representations of the implants to perform planning; classically, plastic overlay foils are manually placed on printed radiographs based on visual assessment. This technique has been applied, e.g. in orthopedic surgery in the preparation of Total Hip Replacement (THR). In recent years, digital planning systems, where digital templates are overlaid to digital radiographs on the computer screen, emerged [Azari & Nikzad, 2008]. With 3D imaging modalities, planning in 3D space became possible. The need for 3D planning has been noticed for several applications, where information from 2D images is insufficient.

Generic Implantation Planning Workflow

The generic workflow for implantation planning is presented in Figure 4.4. To illustrate the procedure, exemplary images are shown from each work step when applied for 2D or 3D planning of THR. It consists of the following work steps:

- Radiological images are selected from a PACS server or storage medium.
- Models of the anatomical structures in the area of implantation are extracted from the images. This step is often omitted in 2D applications.
- The user or an algorithm localizes anatomical landmarks according to which automatic implant selection and placement is performed. Purely manual planning systems often omit this step.
- The user or the system selects an implant or set of implants which optimally match with the anatomical or pathological situation in the patient's body.
- The selected implant(s) is/are registered with patient anatomy.
- Based on the selection and placement of implant templates, the outcome of an implantation can be predicted. For example, THR planning systems often automatically the change in leg length which would result from an implantation plan.

These steps, especially the last three, are usually iterated until a satisfying planning result is obtained.

4.3.1 Overview

DICOM WG24 organized two workshops with representatives from the implant as well as implant planning software industry and clinical experts. Therein, the requirements to the extension of DICOM to include data structures and services for implantation planning were determined with a focus on orthopedic and trauma surgery. An

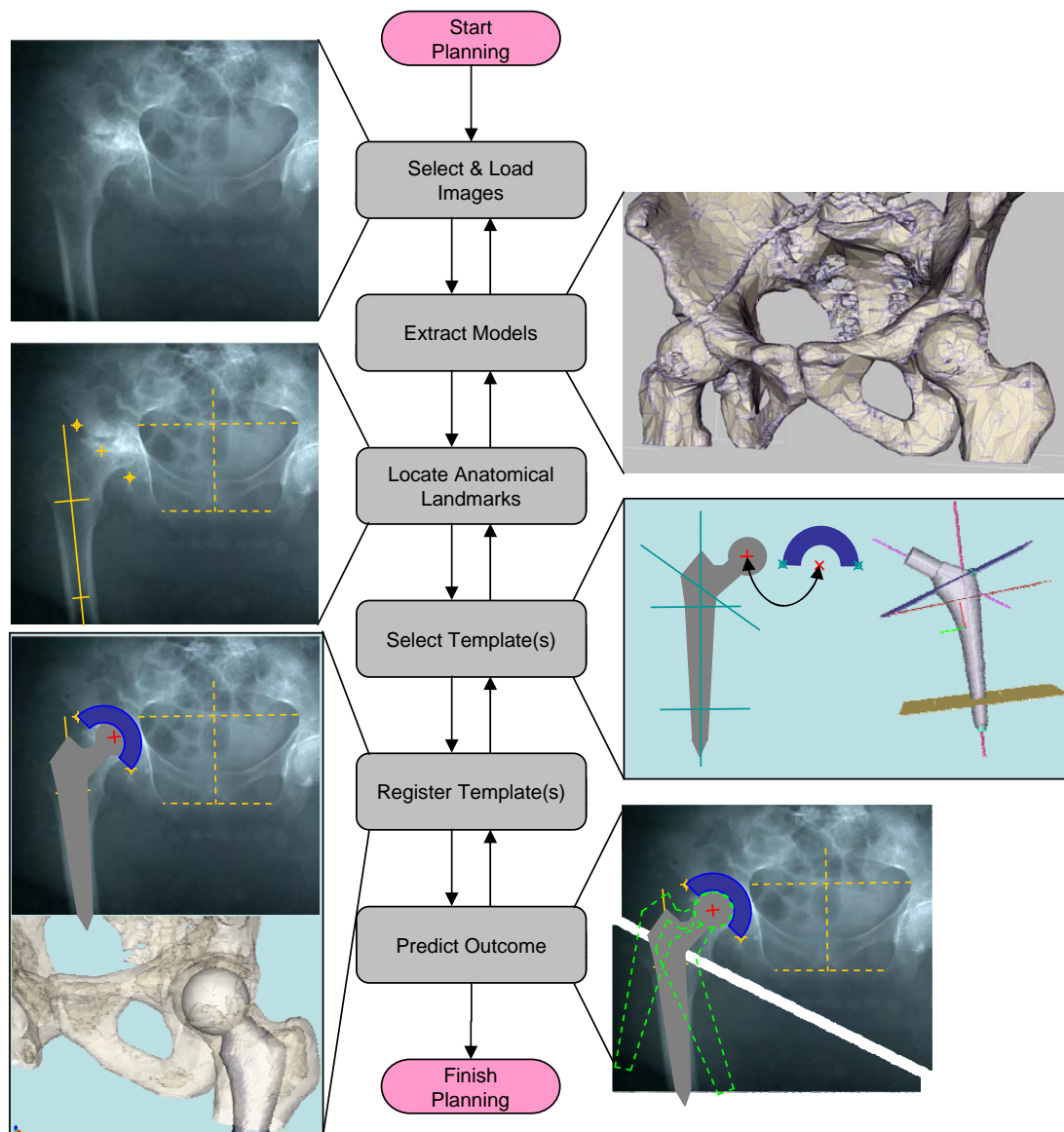


Figure 4.4: Generic workflow for implantation planning.

information model was created which contains four new IEs. The model is depicted in Figure 4.5 According to the model, four IODs and the corresponding services were specified:

- An implant template IOD which represents all properties of an implant which a software or user of a software requires during implantation planning: implant shape, clinical indications, material, ...
- An implant assembly template UID which clarifies the compatibilities between components in modular implants.
- An implant template group IOD which allows to build structured implant template catalogues in order to facilitate browsing through repositories.

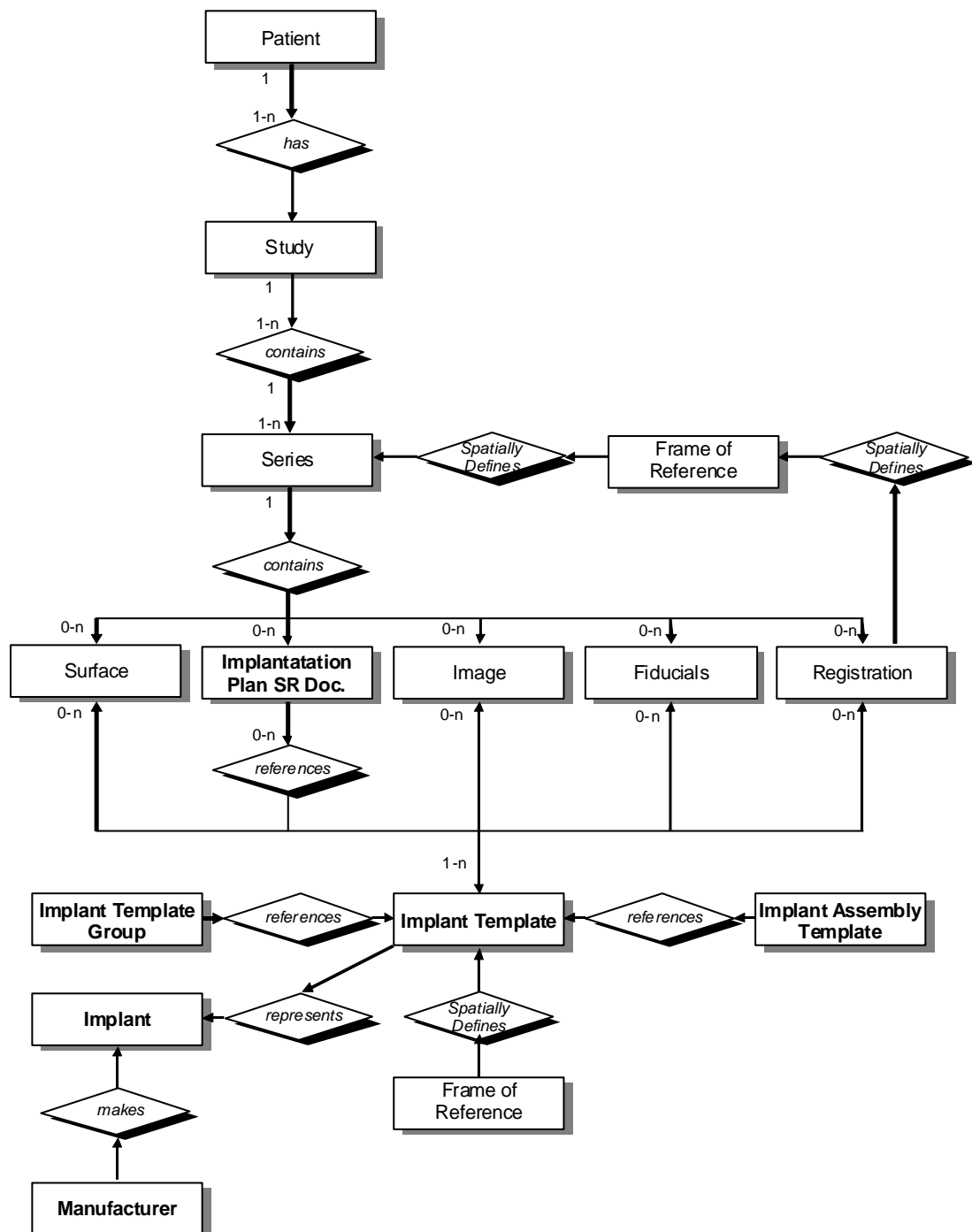


Figure 4.5: DICOM information model for implantation planning.

- An implantation plan SR document which is the patient-specific output of a planning procedure.

DICOM supplement 131 which contains the specifications of the first three IODs. These IODs are the direct result of the presented research work and are presented in the following Sections 4.3.2–4.3.4. The forth IOD is specified in DICOM supplement

134 of which Thomas Treichel from ICCAS, Leipzig, Germany is the principal author. A brief summary of its purpose and structure is presented in Section 4.3.5

4.3.2 Generic Implant Template Storage SOP Class

The generic implant template storage SOP class contains the generic implant template IOD which specifies a data structure for implant templates as they are used in template based implantation planning. Besides the generic SOP common module, which contains the basic DICOM mechanisms for unique instance identification, the IOD contains seven modules (see Table E.8). One of these is the surface module presented in Section 4.2.3. The remaining five modules are specified in supplement 131. A comprehensive list of the attributes in these modules is presented in Appendix E.2. The UML class diagram in Figure 4.6 gives an overview of the IOD and its modules. Each of the modules is presented in detail in the following paragraphs.

Implant Template Description Module

This module contains attributes which identify an implant or describe its purpose and intended use. The complete attributes list of the module is presented in Table E.9. Informative documents and legal notifications addressed to the user of a template can be included in this module as well as a list of countries or regions in which the implant or implant template is not approved for usage. The UML class diagram in Figure 4.7 depicts the information model which was the basis for development of this module. The `ImplantTemplateDescription` class is the central entity of that diagram. It contains a human-readable `name`, the manufacturer assigned `identifier` of the implant the template represents and `size` information.

Implant Template Versions and Derivation Module

Implant templates which are issued by an implant manufacturer or a contracting com-

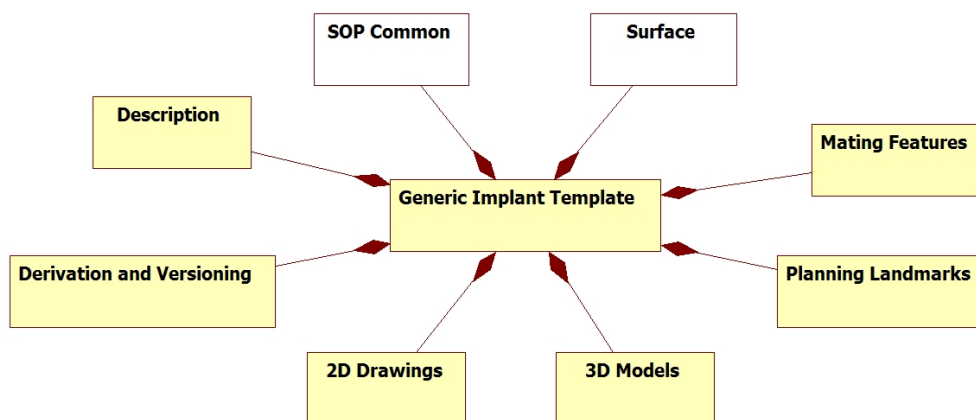


Figure 4.6: UML class diagram of the generic implant template IOD and its modules.

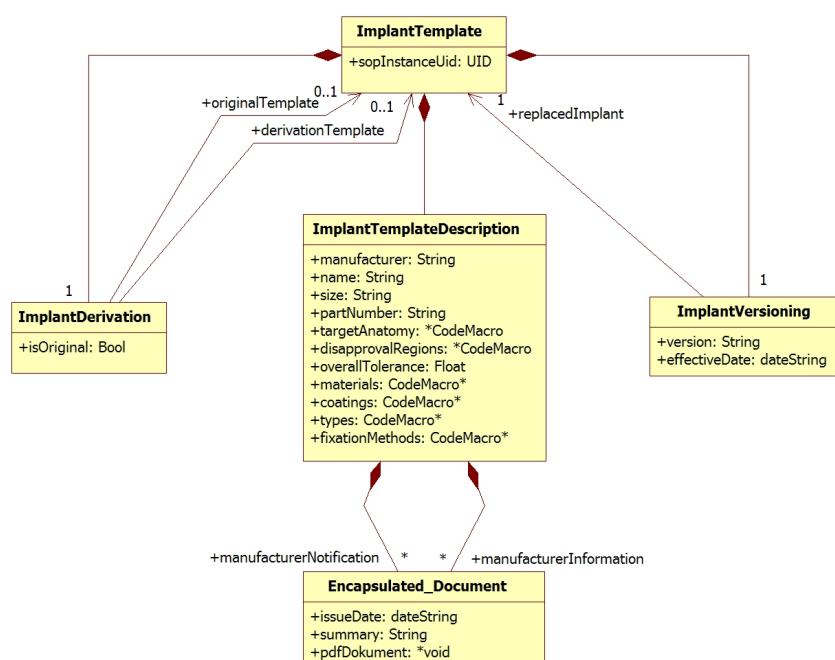


Figure 4.7: UML class diagram of the generic implant template description module and the generic implant versioning and derivation module.

pany are not under all circumstances directly delivered to the health-care providers. Instead, several business cases were discussed during the industry workshops, where the templates are issued through a planning software vendor and/or a PACS vendor or integrator and where each of these entities adds information (in the form of private tags) to the instances. In order to keep control over the origin of all instances and in order to maintain the possibility to exchange plans across systems without exchanging the template repository, references to the original templates from which an instance was derived are included.

In order to fix bugs, to adjust to changing legal regulations, or for marketing purposes, subsequent versions of an implant template can be released. To facilitate version management in the software vendors' as well as end users' systems, a versioning mechanism is provided. Each implant template has a version descriptor. If a preceding version exists, the `ReplacedImplantReference` provides the link to this instance. This attribute can be utilized in a script which replaces outdated versions of implant templates in groups or assembly templates. The attributes of the implant template derivation and versioning module are presented in Table E.10. An example for the reference mechanism included for versioning and derivation is depicted in Figure 4.8.

2D Drawings Module

This module adds the capability to include one or more technical 2D drawings into an implant template instance. Technically, the polyline and line primitives from the DICOM surface module or the DICOM encoding for geometric markups in the DICOM IODs used to describe presentation states would be viable for encoding the 2D

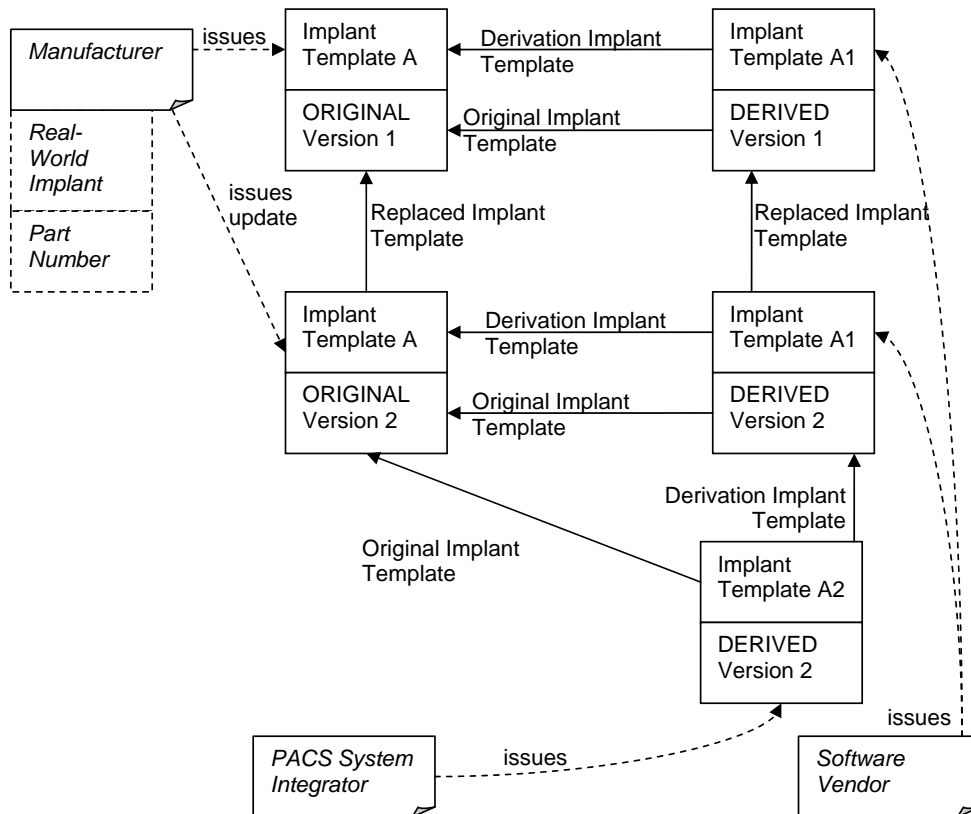


Figure 4.8: Relations between original, replaced, and derived implant templates.

drawings. Nevertheless, the industry representatives who were included in the design process of the implant template IOD incisted on utilizing an encoding which facilitates the export of 2D drawings from the most pervaded computer assisted design software products. The HPGL plotter language was identified as the encoding which best fulfils this criterion.

The implant template 2D drawings module contains a mechanism to include HPGL documents into DICOM instances. One or more documents can be encapsulated in one template instance. The module adds additional attributes to each drawing. These attributes identify the direction of projection and the scaling which was applied to render the drawing. The complete attributes list of the module is presented in Table E.11. Figure 4.9 shows the information model behind this module in a UML class diagram. The green classes are represented by the 2D drawings module.

The `HPGL_Document` class is a container for one 2D drawing. Several of these drawings can be contained in an `ImplantTemplate` instance. Besides the actual `hpglDocument` which is contained as plain byte string, the `HPGL_Document` class assigns an identification number and label to each drawing. The number allows for logical references to a drawing from within or outside the same IOD instance, while the label is intended for display to the human user of a planning software.

HPGL uses integer coordinates which refer to a $\frac{1}{40}$ mm grid on the sheet a plotter would print on. The `scaling` attribute maps *mm*s of the drawing space to real-world

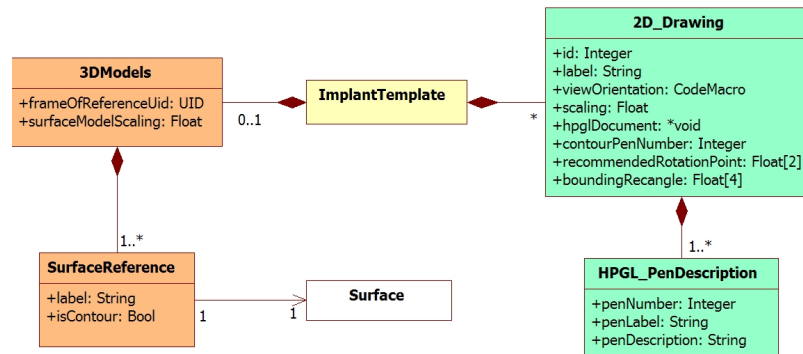


Figure 4.9: UML class diagram of the generic implant template 2D drawings and 3D models modules.

millimeters. To give an example, a line which is 400 units long in HPGL would appear 10 *mm* long in the drawing. If the `scaling` of that drawing would be 2, this line would represent a 20*mm* contour of an implant in the physical world.

In HPGL, multiple pens are used to distinguish between different types of lines, e.g. object contours and axes of symmetry. In HPGL, these pens are identified by an integer ID. The `HPGL_Pen_Description` class adds labels to the pens and distinguishes between contour pens and non-contour pens. Contour pens are used to draw lines which represent the actual shape of the implant. These lines should always be rendered by a planning application, while additional information, such as planning landmarks, lines of symmetry, points of rotation, or dimensioning pens may be hidden.

In many applications, different projective X-ray images are used for planning one case. For each of these projections, a different 2D template is required. It is important, that only 2D templates which represent the view on the implant from the correct direction are overlaid to an X-ray image. To ensure the correct use of the drawings, the class member `viewOrientation` assigns a SNOMED code to the document which describes the direction of projection of the X-ray images which it shall be used with. A similar code is contained in DICOM X-ray images.

3D Models Template

For the geometric definition of 3D templates, the surface module from the surface segmentation IOD (see Section 4.2.3) is re-used. To add semantics to the surfaces in this module, the 3D models module is included in the implant template IOD together with the surface module. The complete attributes list of the module is presented in Table E.12. The orange classes in Figure 4.9 depict the information models behind these modules.

One `ImplantTemplate` object can refer to several `3D_Model` objects. The `3D_Model` class adds labels to surfaces it references through their surface number which is unique within one instance. The `isContour` flag distinguishes between meshes which add to the actual shape of the implant template and should therefore always be rendered and those that add informative content, such as axes of symmetry, directions

of motion or other markups. Since all surfaces defined inside one IOD instance lie by definition in the same `frameOfReference`, the 3D Models module only contains one scaling factor for all surfaces. This factor maps the units in which the points are defined to real-world millimeters.

Planning Landmarks

Planning landmarks serve two purposes: to place and orient implant templates in relation to patient images for overlay visualization and for implant selection based on geometric constraints. A planning landmark is a point, a line, or a plane which is defined in the same space as the implant template. During planning, these geometric features are registered with corresponding landmarks in patient space. Examples for planning landmarks on aortic valve implants are shown in Figure 4.10.

The planning landmarks of an implant template are a triple $L = (L_{pt}, L_l, L_{pl})$ wherein L_{pt} contains the point landmarks, L_l contains the line landmarks and L_{pl} contains the plane landmarks. Every landmark consists of a coded anatomical description (SNOMED codes are emphasized to encode the semantics of the landmarks' anatomical equivalents) and geometric specifications. For 3D templates, point landmarks are represented by one 3D coordinate, line landmarks are encoded as two 3D points which define the line and plane landmarks are encoded as a 3D point on the plane and a 3D normal vector orthogonal to the plane. If the 3D model consists of several surfaces,

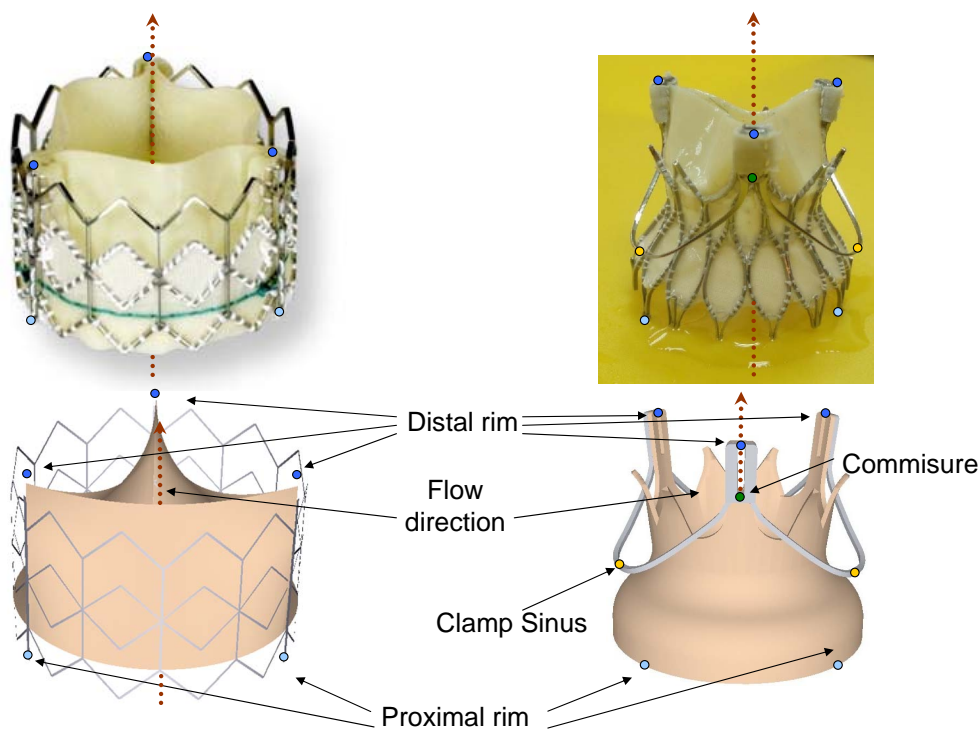


Figure 4.10: Planning landmarks in stented aortic valve implant templates (pictures courtesy of Edwards Lifesciences and Ventor Medical Technologies).

there is still only one 3D definition of each landmark since all surfaces are by definition lying in the same FOR.

For 2D templates, the different projections need to be considered when adding planning landmarks to a template. Each landmark has different coordinates in different projections. Therefore, 2D landmarks are encoded *per projection*: For every projection, one 2D coordinate is assigned to points and two 2D coordinates are assigned to lines. Planes are encoded as lines describing the intersection of the plane with the image plane. It is not required, that every landmark is present in every projection.

The complete attributes list of the generic implant template planning landmarks module is presented in Table E.14. It is represented by the yellow classes in Figure 4.11.

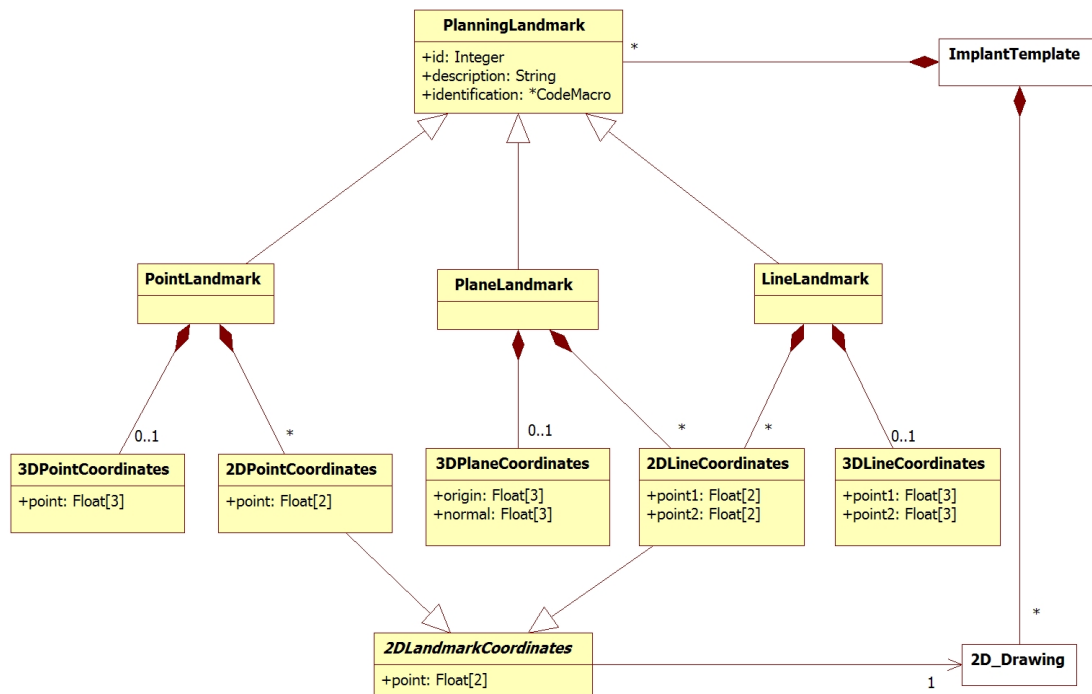


Figure 4.11: UML class diagram of the generic implant template planning landmarks module.

Mating Features

Implants can consist of several components. Usually, implant components are constructed in a way that when two components are assembled, they "snap" into one specific alignment. This alignment can have degrees of freedom to realize joints.

The implant template mating feature module contains an ordered set $S = \{S_1, \dots, S_k\}$ of ordered sets $S_i = \{M_1, \dots, M_l\}$ of *mating features* M_j . Each mating feature M_j is described by a 4×4 Matrix $M_j = (a_{jx}, a_{jy}, a_{jz}, o_j)$, containing three orthogonal axes a_{jx}, a_{jy}, a_{jz} and one origin o_j in homogeneous coordinates. Each mating feature specifies a local coordinate system. The attributes required to describe these sets

are contained in the generic implant template mating features module (see Table E.13) which is represented by the yellow classes in Figure 4.12. Similar to planning landmarks, mating features require a geometric definition in every 2D drawing according to the angle of projection and scaling of that drawing.

When two components are assembled, the templates are arranged so that the origins and axes of the mating features coincide. Joints can be specified by adding degrees of freedom to mating features. The mating features are organized in sets to allow the following restriction: In a valid assembly, at most one mating feature M_j of each mating feature set S_i of each participating component may be used.

Mating features are a tool to constrain assembly of components geometrically. In order to build useful assemblies, an assembly template is required which contains the information about which component can be assembled with which component. For that purpose, a separate IOD was developed which is described in the following section.

4.3.3 Implant Assembly Templates

Implants are often not monolithic objects but consist of several components which can be selected from several sizes and shapes. The components are assembled before or during implantation. For implantation planning, two aspects of component assembly are relevant. Firstly, a planning system has to know which combinations between components are allowed and which not. Secondly, the planning system has to know the geometric relation of the components after assembly.

The implant template IOD contains the definition of available mating features on im-

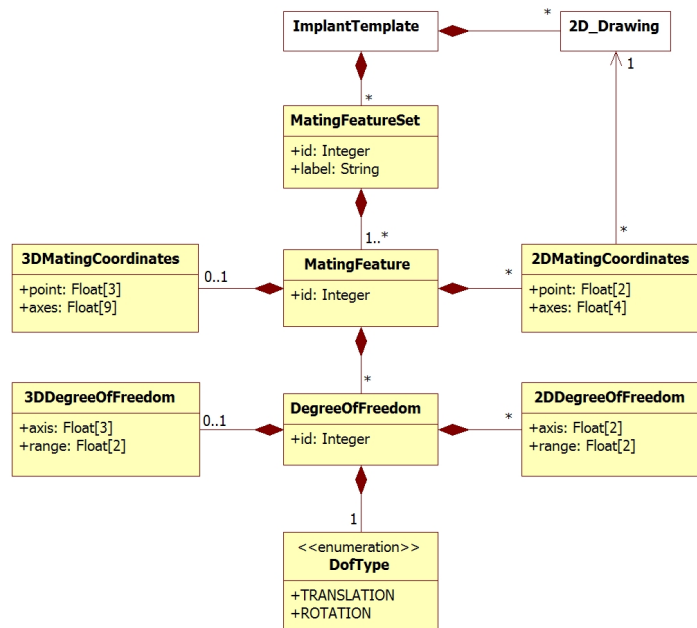


Figure 4.12: UML class diagram of the generic implant template mating features module.

plants. These features do not contain compatibility information, e.g. the mating point on the cone of a hip shaft contains no indication about its destiny to mate with a hip joint head and its impossibility to mate with the screw hole in a trauma plate. Assembly templates refer to the implant templates for which they describe possible assemblies. To use an analogy, implant templates define the Lego[®] blocks in a construction kit while the implant assembly template represents the construction manual. The mating features in the implant templates are used to clarify how the components are to be assembled.

An implant assembly template A is a pair $A = (C, F)$ consisting of an ordered set $C = \{C_1, \dots, C_n\}$ of references to implant templates and a set of sextuples $F = \{(c_1, i_1, j_1, c_2, i_2, j_2)\}$. Each element of F references exactly two implants $C_{c_1}, C_{c_2} \in C$ and exactly one mating feature $M_{j_1,2}$ from exactly one mating feature set $S_{i_1,2}$ in each implant. Each element of F specifies one *possible* connection between the referenced components C_{c_1} and $C_{c_2} \in C$ using mating feature M_{j_1} from C_{c_1} 's mating feature set S_{i_1} and mating feature M_{j_2} from C_{c_2} 's mating feature set S_{i_2} . All indexes are required to refer to elements existing in the sets.

An implant assembly is a set $\hat{A} = (\hat{C}, \hat{F})$, where $\hat{C} = \{C_1, \dots, C_n\}$ is an ordered set of references to implant templates and $\hat{F} = \{(c_1, i_1, j_1, c_2, i_2, j_2)\}$ is a set of sextuples referencing exactly two implants $C_{c_1}, C_{c_2} \in C$ and exactly one mating feature from exactly one mating feature set in each implant. Each element of \hat{F} specifies one *actual* connection between the referenced components C_{c_1} and $C_{c_2} \in C$ using mating feature M_{j_1} from C_{c_1} 's mating feature set S_{i_1} and mating feature M_{j_2} from C_{c_2} 's mating feature set S_{i_2} . All indexes are required to refer to elements existing in the sets. A *valid* implant assembly is an implant assembly where no more than one mating feature is used in each mating feature set:

$$\begin{aligned} \exists f_1 &= (c_1, i_1, j_1, c_2, i_2, j_2) \in \hat{F} \rightarrow \exists f_2 \neq f_1 : \\ &f_2 = (c_1, i_1, j_\alpha, c_2, i_\gamma, j_\beta) \forall j_\alpha, j_\beta, i_\gamma \vee \\ &f_2 = (c_1, i_\delta, j_\alpha, c_2, i_2, j_\beta) \forall j_\alpha, j_\beta, i_\delta . \end{aligned}$$

The spatial relation between two implants C_{c_1} and C_{c_2} that are directly assembled can be calculated as a rigid transformation using affine matrices:

$$T_{1,2} = M_{c_2} \cdot M_{c_1}^{-1}$$

Implant Assembly Template Storage SOP Class

The implant assembly template SOP class consists of the implant assembly template IOD and the services necessary to store, query and retrieve instances of that IOD. The IOD contains information describing the issuer and the clinical purpose of the template, as well as sequences encoding the ordered sets described above. One implant assembly template IOD instance is able to represent all combinations of components which the issuer finds apt in order to fulfill a clinical purpose.

The UML class diagram in Figure 4.13 shows the information which is contained in an implant assembly template IOD. The IOD contains the SOP common module for unique instance reference and the implant assembly template module which is presented in Table E.19. `implantAssemblyTemplates` are specific to one `procedureType` and `surgicalTechnique`. The anatomical structure(s) to which the procedure can be applied is listed in `targetAnatomy`. Standardized codes, such as SNOMED CT, are defined to encode these values.

The components which are used in the assemblies specified in one instance are organized in `ComponentGroups` which assign roles to components. Some components, such as a screw or plate, are generic enough to play several roles and therefore can be part of more than one component group. Besides a `label`, a component group has two flags to indicate whether during planning at least one representative of a group needs to be selected (`isMandatory`) and whether or not only one representative of that group may be selected (`isExclusive`). For example, in a multi-component hip joint assembly, the group of hip stems would be labeled both, mandatory and exclusive: Exactly one stem component is required in a valid assembly.

The implant assembly template storage SOP class consists of the implant assembly template IOD and the services to store, query, and retrieve instances of this IOD via a PACS network. The IOD consists of the SOP common module and the implant assembly template module which is specified in supplement 131. It can be looked

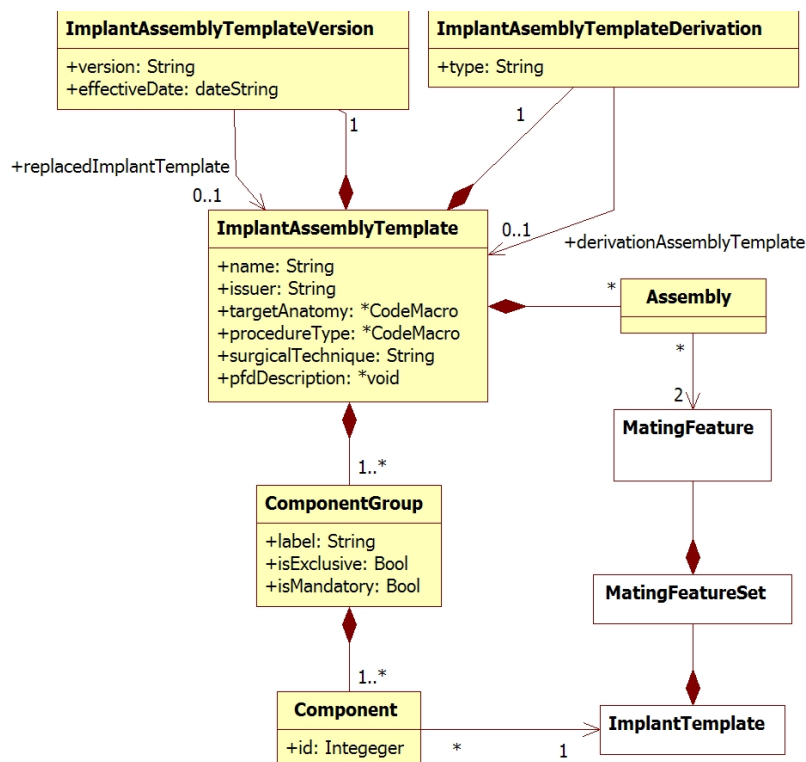


Figure 4.13: UML class diagram of the implant assembly template module.

at as a serialization of the `implantAssemblyTemplate` class and all classes this class aggregates. In Figure 4.13, these are the yellow classes.

In Figure 4.14, four implant templates are shown. The table in the figure represents the relevant information contained in an assembly template. The implant to the left of the image, a hip stem implant, contains three mating features (abbreviated MF) in one mating feature set (abbreviated MFS). The three implants to the right, femoral head implants in different sizes, have one mating feature in one mating feature set each. The implant assembly template defines two component types, both mandatory and exclusive. The assembly sequence contains three items, specifying the possibility to mate the stem with each head. Thereby, each head connects to a different mating feature on the stem. The black double-pointed arrows in the image illustrate the specified assemblies.

4.3.4 Implant Template Groups

In applications such as trauma surgery, the catalogue of implant templates from one vendor can easily mount up to contain several thousand parts. In order to organize such vast numbers of objects and in order to facilitate browsing through the catalogue, a grouping mechanism is provided by the implant template group IOD (see Appendix E.4 for complete IOD and module definition).

With this mechanism, any number of implants can be referenced and ordered according to several dimensions. In the case of metal plates as they are used in trauma surgery, dimensions according to which the implants are ordered could be, e.g., length, thickness, and number of screw holes. The operator of a planning workstation can browse through the catalogue along these dimensions with the implant templates being visualized in patient space. Thereby, it is required that the templates keep their alignment with patient anatomy. Technically, it is possible that templates from different product lines or manufacturers are in one template group. It is possible (or, by experience, likely) that these templates are defined in different frames of reference. The implant template group provides a registration between these FORs to prevent unexpected "jumps" when switching between group members: each group specifies a local coordinate system; for each template in the group, the group specifies one 2D coordinate system in each HPGL document and one 3D coordinate system in the surface representation which represents the group's coordinate system in this template. Similar to mating features, the group coordinate system can be used to align the templates from one group. An example for an implant template group is shown in Figure 4.15. The figure shows four implant templates which describe different hip stem components. Each template is drawn according to its own local coordinate system (red). The group definition adds a common reference coordinate system (green) to each drawing, according to which the stems can be aligned (right image).

Figure 4.16 depicts the information model behind the implant template group module. A `ImplantTemplateGroup` aggregates an arbitrary number of `GroupMember`

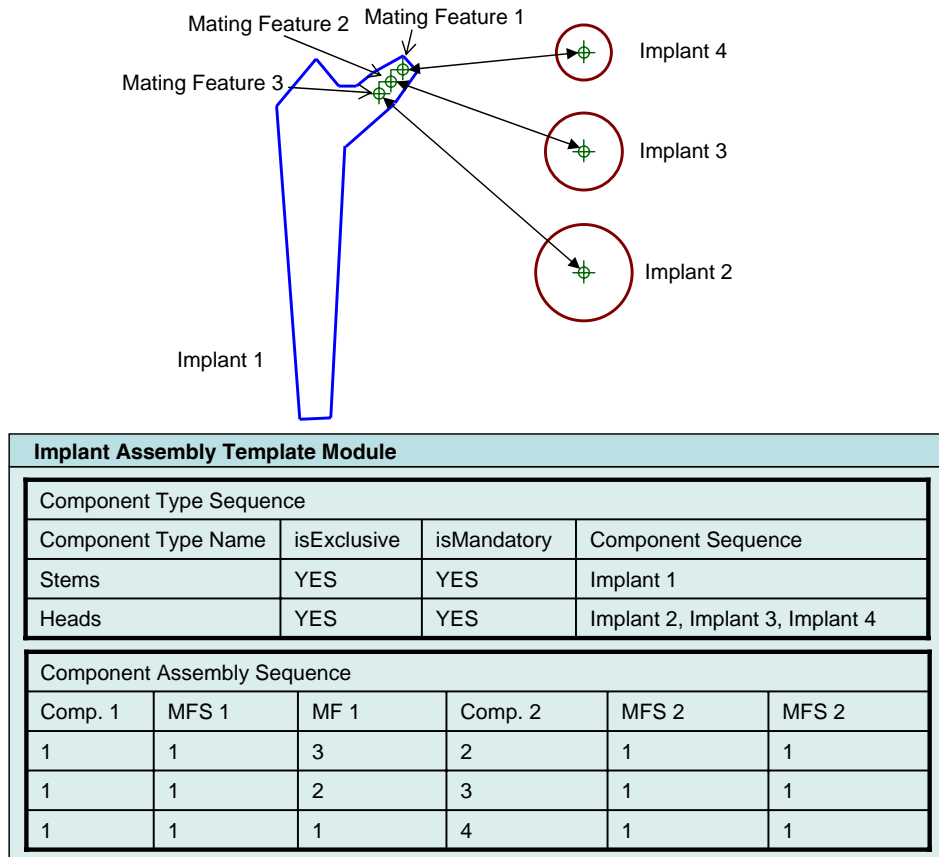


Figure 4.14: Implant templates, mating features, and an implant assembly template.

objects. These contain a reference to an external `ImplantTemplate` and the specification of the 3D reference coordinate systems and 2D reference coordinate systems.

The `VariationDimension` class contains the attributes which are required to implement the multidimensional ranking mechanism. Each variation dimension has a name to identify it and contains a list of `Rank` objects which assigns a rank to a `GroupMember`. Several members can have the same rank in one variation dimension, such as, for example, several trauma plates can have the same number of screw holes.

4.3.5 Implantation Plan SR Document

The implant template IOD and its adjunct IODs, the implant assembly template IOD and implant template group IOD contain information about implants which are not specific to a patient. The information which is encoded in these IODs describes the shape and possible or intended usage of implants. For patient specific implantation plans, a fourth data structure and additional services are required.

Complementing supplement 131 "Implant Templates", work has been started on

supplement 134 "Implantation Plan SR Documents". With this supplement DICOM Working Group 24 attempts to standardize the encoding of implantation planning results. This data structure is based on the Structured Reporting (SR) mechanisms [Treichel *et al.*, 2010] in DICOM which were originally introduced for the encoding of diagnostic findings related to radiology images.

Figure 4.17 shows the structure of the proposed DICOM implant plan SR document. The document itself contains only little information: meta data referring the plan to a patient and scheduled day of intervention and some descriptive text and comments. In addition to this, the plan makes intensive use of the composite instance reference mechanism which allows an SR to refer to DICOM SOP instances. An implant plan SR document can refer to:

- the images and surfaces of the patient which were utilized during planning,
- the fiducials which were generated or utilized during planning,
- the implant templates which were selected during planning,
- the spatial registration instance which contains the positions and orientations of all implant templates in relation to the frame(s) of reference of the patient images and surfaces,
- screenshots, and visualizations of the plan.

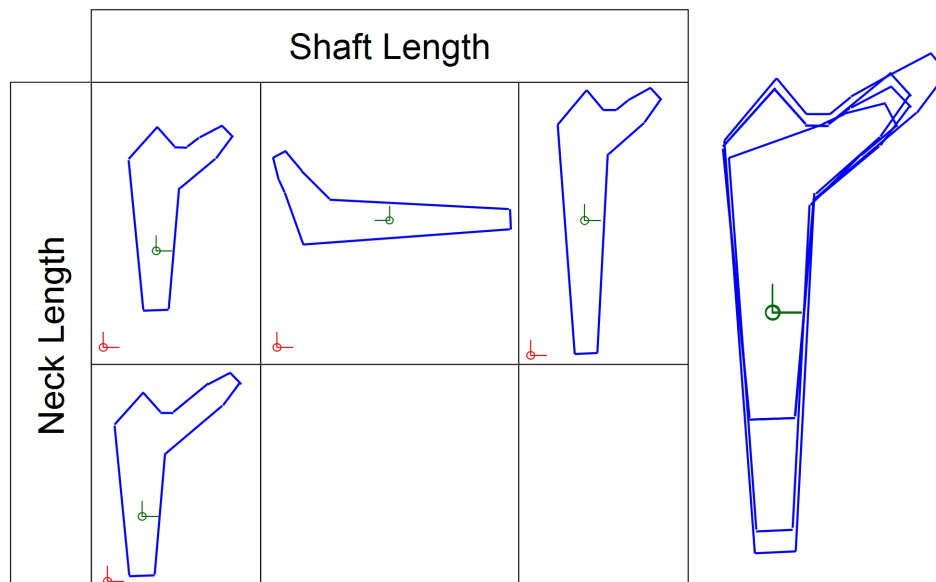


Figure 4.15: Implant template group. Left: Four stem components are ranked according to stem length and neck length. Each drawing space has its own origin (red coordinate system). The group adds matching coordinate systems (green) which can be used to align the templates during planning (right image).

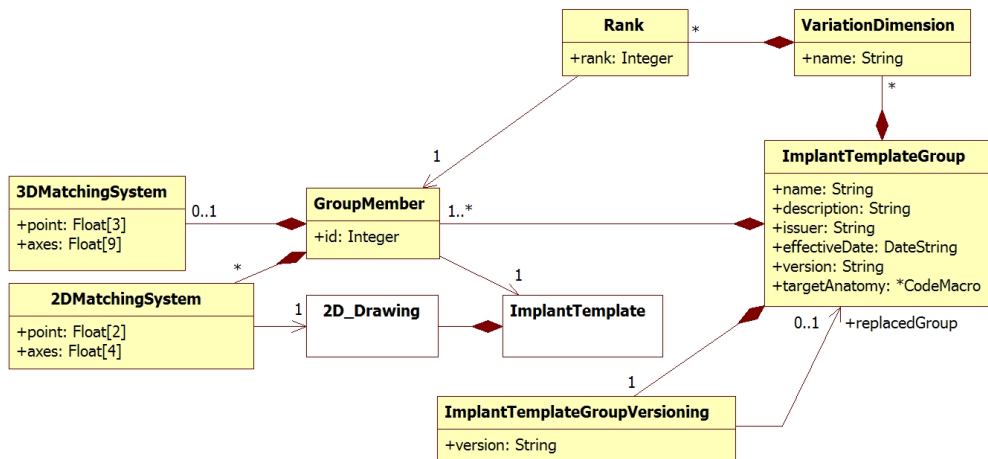


Figure 4.16: UML class diagram of the implant template group module.

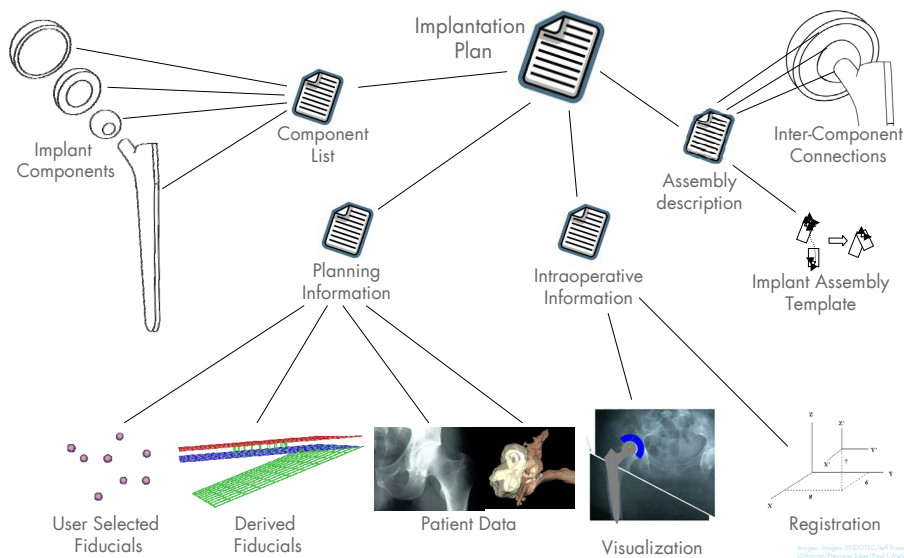


Figure 4.17: Structure of a DICOM implant plan SR document (Image courtesy of Thomas Treichel, Universität Leipzig, Germany).

4.4 Summary

Two DICOM work items were presented which affect the exchange of surgical planning results. The surface segmentation SOP class was developed as a means to store and exchange geometric information which is extracted from patient images. It adds a new class of instances, surface meshes, to the DICOM standard. This was, for two rea-

sons, an important prerequisite for the integration of surgical planning scenarios into a PACS environment. Firstly, surface segmentations are frequently used in CAS applications during planning as well as intraoperatively. The surface segmentation IOD enables the transfer of segmentation results from a planning workstation to an intraoperative system via the PACS server. Secondly, with the surface segmentation IOD the surface module was added to the DICOM standard. Future work items in surgery and other fields of application which include the description of geometric content are facilitated by this module.

The implant template SOP classes were proposed as a vendor-independent means to encode template models of implants as they are used in implantation planning. The supplement aims at reducing the effort for a software vendor to include templates from different implant manufacturers into planning software as well as at enabling the data exchange between a planning workstation and an intraoperative assistance system through the PACS server as the centralized database for patient images and related information. The implant template SOP classes are closely related to the implantation plan SR document, which is a patient-specific file containing the results of a planning procedure.

Chapter 5

An Open-Source Interface for OR Integration

The development of distributed CAS systems consisting of independent modules for specific functionalities requires an infrastructure through which the modules exchange data, commands, status messages, and other notifications. In this chapter a software library is presented which acts as a *surgical middleware* [Cleary & Kinsella \[2004\]](#) according to the following functional requirements:

- **Message exchange:** A messaging service is required through which systems are able to exchange notifications, status messages, or larger data sets such as images or models. The message service may not be scheduled, i.e. a device must be allowed send a message to another device at any time without having to wait for a time slot. The only restriction to the frequency, size, and time at which messages are sent are to be the limitations which are owed to network bandwidth.
- **Data streaming:** For continuous exchange of data, such as transmission of videos or biosignals, a streaming service is required. In contrast to the messaging service, streaming connections are based on a service-level-agreement which is negotiated during the initialization of a session. This agreement between the sender and the receiver of a data stream regulates the number of frames which are sent per second and the size of each frame. A module which accepts a request to send a data stream of a certain frame rate and frame size guarantees that it will sustain the data stream according to these parameters or at least send a notification to the subscriber(s) of a stream when it can no longer sustain it.
- **Read- and write-access to device parameters:** A mechanism is required which enables a system to grant peer systems read-only or read-write access to its internal parameters. This functionality is required to realize centralized display of device states and remote control over the parameters of a device.
- **Access to methods:** An interface is required through which one system can call a method or function in another system. This interface requires a mechanism for the exchange of input parameters and results.

- **Surgical Plug-and-Play:** Distributed systems require intensive setup activities in order to establish the connections between modules. An auto configuration mechanism is required through which modules can discover and identify each others and exchange information about their functionalities.

The following non-functional requirements are imposed to an infrastructure which implements these functionalities:

- **Thread safety:** The implementation of these services has to allow for accessing these services from several threads within the application that runs the interface as well as from several clients simultaneously. Critical code segments have to be secured with semaphores or other techniques to prevent two threads from accessing the same memory area or network socket at the same time.
- **Stability:** The interface is built to be immune to denial of service attacks. Especially in server application, there exists the risk that multiple clients request services at the same time, flooding the interface with more requests than the server can handle. Applications must be able to set a limit to the number of requests they process at a time. The interface through which requests are received must ensure that this limit is not exceeded.
- **Performance:** The delay and throughput of message and stream exchange shall only be limited by the underlying network. For status messages, a latency of up 20 milliseconds is acceptable.
- **Portability:** No libraries or programming concepts shall be utilized which restrict the compatibility of the implementation with any of the following operation systems: Windows XP, Windows 2003 Server, Linux, MacOS.
- **Maintainability:** The Plug-and-Play mechanism must include a human-readable name to every device and every service offered by a device to facilitate the required user interaction during setup of a system.

5.1 TiCoLi - An Overview

The TIMMS Communication Library (TiCoLi) is a c++ class library. It was developed with the aim to provide an open source class library for peer-to-peer communication between modules in a distributed CAS system. Software which includes the TiCoLi library can use its API in order to act as a server and as client for message exchange, attribute and method access, and streaming.

The communication between TiCoLi devices (i.e. devices which include the TiCoLi and use it to cooperate with other TiCoLi devices) is based on peer-to-peer sessions. Each session opens one TCP connection exclusively used for exchange of messages and commands for this session. For data streaming, additional UDP sockets are reserved per stream. Once a session is established, both systems can use all services of

other system, there is no clear assignment of a server and a client role for one device. In the following text, the terms *server* and *client* are always used in the context of one service invocation. The term *server* is used to identify the application which offers a service, i.e. can send streams or grants access to its internal attributes or methods. A *client* is a device which intends to invoke any of these services in a server. The same device can be a server with regard to one service and at the same time be the client with regard to another service. For example, a device could be the client of a server which broadcasts a biosignals, apply signal processing to it, and act as a server which sends notifications to its subscribers when one signal exceeds a certain threshold.

The TiCoLi is based on several standardized protocols and open source libraries. Figure 5.1 shows the TiCoLi protocol stack. The TiCoLi provides an application-level interface to the functionalities of the underlying protocols. The protocols and toolkits are described in Appendix C.

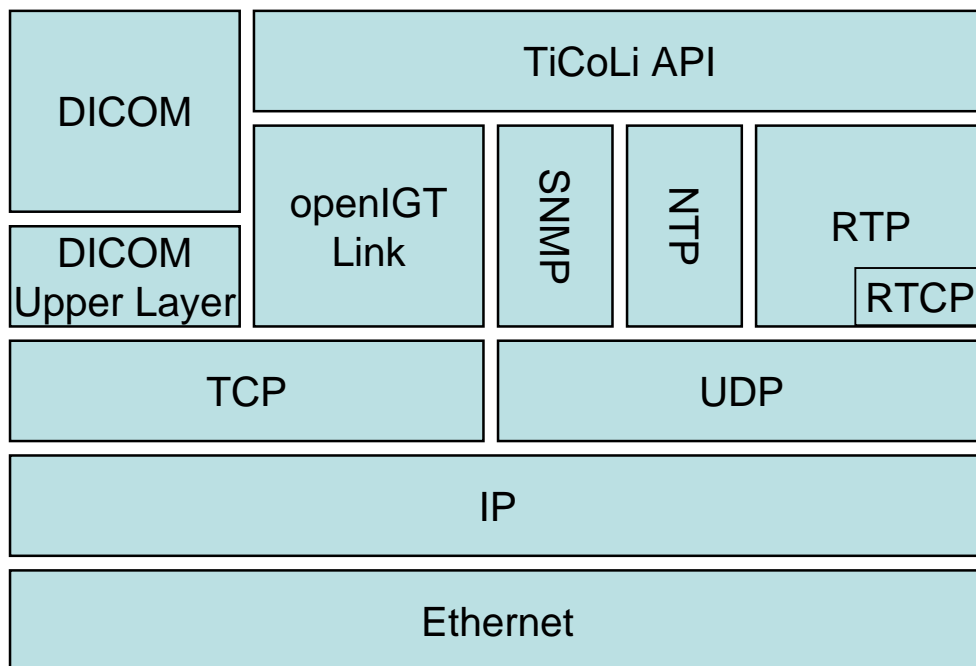


Figure 5.1: TiCoLi protocol stack.

5.2 TiCoLi: Basic Types

The TiCoLi defines a number of basic types which are used in many of the interface functions of the API and appear in all the public classes of the library. Of these types and classes, `Condition`, `Handle`, `HandleSet`, and `Callback` will appear in the following sections and are therefore introduced before the functional modules are described.

Condition

`Condition` is an enum type definition which is used as return parameter for most interface methods of the TiCoLi API. The `Condition` an interface method returns is either OK if the method could successfully be executed or otherwise an appropriate error code.

Handle and HandleSet

`HandleSet` is a template container class which is used internally by the TiCoLi to organize information about the services a device offers as well as about peer devices and the services they offer. `HandleSet` assigns a random integer `Handle` to each device it contains. The `Handles` are used by an application to select services or devices when calling interface methods on the TiCoLi API. The `HandleSet` class and its instances can directly be reached from the application. Instead, the `HandleSets` are managed internally by the member classes of the TiCoLi API.

Callback Functions

Callback functions are used by the TiCoLi to send notifications to the application which initialized the TiCoLi. For example, in order to be able to receive messages through the TiCoLi, an application assigns a pointer to a global function or member function which the TiCoLi API shall call every time a message is received. Several kinds of callback functions exist:

The abstract template class `CallbackBase<class P>` provides the generic interface to a pointer to a function with the signature `(P, Handle)`. It inherits `CallbackThreadCounter` and adds the definition of a method `Call(P value, Handle handle)` which is called from the TiCoLi to call the method contained in a `CallbackBase<class P>` instance. `Call(...)` tries to increment the thread counter before generating a `CallbackThreadData<P>` instance and creating a thread from which it executes the actual function the function pointer refers to. `Call(...)` is thread safe.

`CallbackGlobal<class p>` inherits from `CallbackBase<class P>` and is a concrete implementation of the callback pattern for global functions. It contains a pointer `mFuncPtr` to a global function with the signature `(P, Handle)`.

`Callback<class T, class P>` is a concrete implementation of the callback pattern for class member function. It inherits from `CallbackBase<class P>` and contains a pointer `mInstance` to an instance of the class `T` and a pointer `mFuncPtr` to a member function of this instance with the signature `(P, Handle)`.

`VoidCallbackBase`, `VoidCallback`, and `VoidCallback<Class T>` are specialized implementations of the callback pattern for function pointers to global or member functions with no arguments.

In order to allow multiple threads to use a callback function simultaneously and at the same time prevent an application from being overrun by function calls, the callback classes contain a mechanism for thread counting. Every time the callback is called, a

thread is started in which the function pointer is called and a counter is incremented. When a thread is completed, the counter is decremented. A threshold is assigned to every callback function which limits the number of threads that are allowed to run the function at a time. When this number is exceeded, a `BUSY` condition is returned. POSIX thread mutexes are utilized to prevent threads from slipping through this limit by ensuring that invocations of callback threads are handled one at a time.

The processing of callback calls is presented in Appendix [D.2](#).

5.3 The API, the Core, and the Managers

The TiCoLi API is based on a modular system design (see Figure [5.2](#)) with one central mediator module, the `TiCoLiCore` and five modules which implement the five services of the TiCoLi: autoconfiguration, message exchange, attribute access, remote method calls, and data streaming. Each module contains a manager class which implements most of the behavior of the module (see Sections [5.3.1](#) – [5.3.5](#)).

Against an application which uses the TiCoLi, the whole library is hidden behind the `TiCoLiAPI` class which offers an interface to all functionalities through static methods.

The `TiCoLiCore` holds a static object of itself and aggregates static objects of the five `Manager` classes. The core is the central distributor for messages which are exchanged between the managers, especially when external commands received from peer devices are processed. The interaction of the `TiCoLiAPI`, the `TiCoLiCore` and the `Managers` is presented below where the interface methods of the different modules are presented.

5.3.1 The Device Manager

The device manager holds all information about the services an application is offering through the TiCoLi. In addition, the device manager obtains and stores information about peer TiCoLi devices in the network (see Figure [5.3](#)). The `DeviceManager` stores this information in a `HandleSet` containing objects of the `DeviceDescriptionInternal` class. Whenever a component of the TiCoLi or the application which uses the TiCoLi requires information about an application in the network, it accesses the device descriptions which are managed by the `DeviceManager`.



Figure 5.2: UML class diagram of the TiCoLi API, core, and managers.

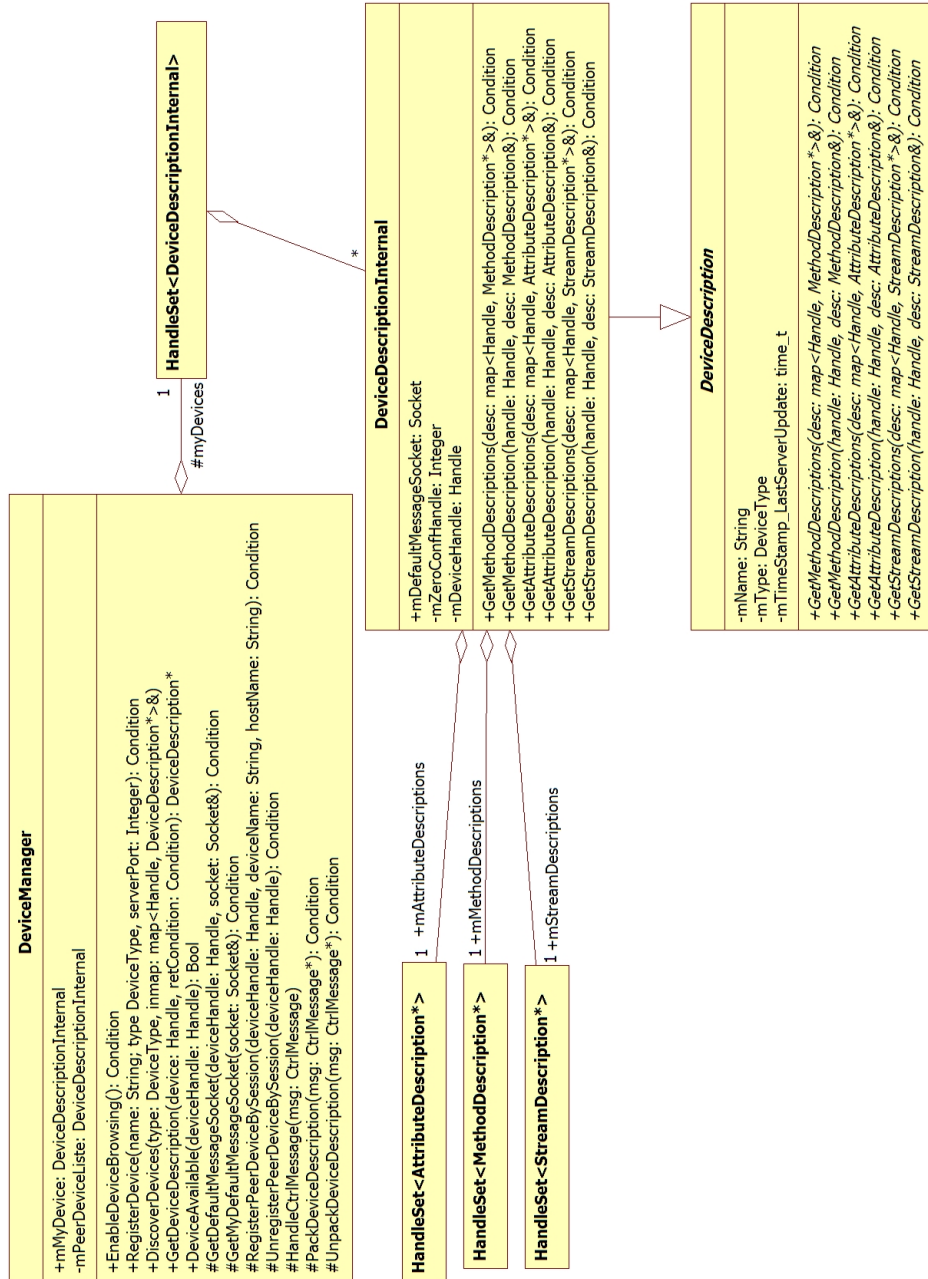


Figure 5.3: UML class diagram of the TiColi DeviceManager and related classes.

Internal and External Device Descriptions

Device descriptions are used to identify a device in a network. The device description of a TiCoLi device contains a unique name, a type, and information about the network socket through which a connection can be established. This information is exchanged among TiCoLi devices using the zeroConf service (see Appendix C) which runs in the background. Two classes are present in the TiCoLi to represent the descriptions. `DeviceDescription` contains only a subset of the information and is handed out to an application which requests a list of available peers through the TiCoLi API. This abbreviated description omits all technical details in order to hide them from the application. Each instance of `DeviceDescription` contains only the name and type of one peer device together with a time stamp of its last update.

The device type is an `enum` for which the TiCoLi defines the following values:

- `DEV_UNKNOWN`: Default value assigned during instance creation.
- `DEV_TCC`: The TIMMS Component Controller is a central device in an OR network which manages the available bandwidth, monitors network and system activity, and can act as a watchdog which queries the system state of all devices to identify technical problems.
- `DEV_VIDEO`: Any kind of video source, be it a camera or other imaging modality, a video processor, or a device which replays recorded videos.
- `DEV_BIOSIGNAL`: Any kind of biosignal source, i.e. a device which measure directly or receives, processes, and broadcasts data about the physical state of the patient, e.g. an ECG device.
- `DEV_TRACKING`: All kinds of devices which generate or receive, process, and broadcast tracking data, e.g. a tracking camera or a stereotactic arm.
- `DEV_CLIENT`: Assigned by devices which receive a connection request from a peer which does not publish a device description through zeroConf.

`DeviceDescriptionInternal` is used internally in the TiCoLi to represent all properties of a peer device. The `DeviceManager` holds a `HandleSet mPeerDeviceList` which contains the descriptions of all known peer devices. Besides the `mPeerDeviceList`, the `DeviceManager` contains an instance `mMyDevice` which contains the name, type, and ports of the local device.

`DeviceDescriptionInternal` has the following member variables:

- `string mName`: The name of the device (inherited from `DeviceDescription`)
- `DeviceType mType`: The type of the device (see above, inherited from `DeviceDescription`)

- `time_t mTimeStamp`: Time of the last update of this description (inherited from `DeviceDescription`)
- `Handle deviceHandle`: Identifies the device locally. This handle is handed out to the application to identify the peer device when calling services. Each TiCoLi implementation in a network assigns its own `Handles`, i.e. the same device can have different `Handles` in the lists of its peer devices (see Figure 5.5).
- `Socket defaultSocket`: The socket (IP address + TCP port) at which the device accepts connection requests.
- `int mZeroConfHandle`: The identification number the zeroConf implementation assigned to the device.¹
- `const char *hostname`: The DNS name or IP number of the device.

Interface Methods

For interaction with the device manager, two methods are defined in the TiCoLi API header.

`Condition RegisterDevice(...)` is called by a server application to publish information about itself in the network. A name and type have to be assigned. The `serverPort` identifies the TCP port which shall be opened by the TiCoLi API for connection requests. If the method is called with an invalid `serverPort`, the next free port starting from 10,000 is selected. The method returns a `Condition` which is OK if the device could be registered and its presence announced to the network. Otherwise an appropriate error code is returned. To withdraw all services from the network, a server application calls the `Condition UnRegisterDevice()` method.

In order to obtain a list of registered peer devices, an application calls the `Condition DiscoverDevices(...)` method. An empty `std::map` container is handed to the API by the application. The `Handles` and `DeviceDescriptions` of all known peer devices are filled into the container. The method returns OK if the map could be compiled and an appropriate error code otherwise.

Interaction with the Bonjour Service

The `DeviceManager` interacts with the Bonjour implementation of the zeroConf standard to broadcast and receive multicast DNS/DNS-SD requests (see Appendix C). The inclusion of the zeroConf standard in the TiCoLi is based on preliminary work of Stefan Bohn at ICCAS who began to experiment with the idea of zeroConf for

¹Both the zeroConf handle and the TiCoLi device handle uniquely identify a device. This redundancy is owed to the fact that the zeroConfHandles are not compatible with the `Handles` used by the `HandleSet` class. Since both handles only use up 32 bits of memory per device, this redundancy was regarded acceptable and preferred to the inconsistency of using a different container class for device descriptions than for anything else.

device discovery in the OR after the zeroConf standard was added to the normative references of the DICOM standard with Correction Proposal (CP) 633 in 2006.² When initiated, the TiCoLi sends a multicast service discovery request to all computers in the network. All TiCoLi devices which offer any TiCoLi services reply to the request with a DNS-SD response. The device type "TiCoLi" is used to identify TiCoLi devices. From these answers, the requesting TiCoLi instance compiles an initial list of known peer devices. The `DeviceManager` creates instances of `DeviceDescriptionInternal` accordingly and stores them in the `mPeerDeviceList` class member.

When an application calls `RegisterDevice(...)`, its `DeviceManager` uses the Bonjour service to send a multicast message to announce its existence to the peers. The zeroConf service running in all TiCoLi devices will receive this notification and send a notification to the `DeviceManager` which updates its `mPeerDeviceList` accordingly.

The processing of zeroConf events is performed by the `DeviceManager` in the background and is completely transparent for the application. In order to retrieve an actual list of peer devices, the application has to call `DiscoverDevices(...)` on the TiCoLi API. The UML sequence diagram in Figure 5.4 shows a case where a device announces its presence to the network and is added to the `mPeerDeviceList` of a peer device.

- During initialization of the TiCoLi, the API starts the zeroConf service by calling `EnableDeviceBrowsing()` on the `DeviceManager dm1`. (1)
- `dm1` starts the browsing service in its `zeroConf` instance `z1` (2). The DNS request for active peers remains unanswered, since no peers are present at this point of time (3, 4).
- A background thread is started which continuously browses for events sent from the zeroConf service (5).
- A second application is started and registers a device. The call is forwarded to the `DeviceManager dm2` (8), which initializes its `ZeroConf` instance (9) and creates an instance `ddi2` of `DeviceDescriptionInternal` to which it assigns the parameters with which `RegisterDevice(...)` was called (10, 11).
- The `DeviceManager dm2` activates the `MessageManager mm2` which opens the default server port to enable reception of connection requests (13 – 16).
- The `dm2` activates the announcement of the device description by calling `RegisterDevice(...)` on its `ZeroConf` instance (17).

²CP 633 declares the utilization of multicast DNS-SD for the exchange of application entity titles, network sockets and primary device type (e.g. "archive", "image capture", or "film digitizer") for the establishment of *ad-hoc* DICOM networks.

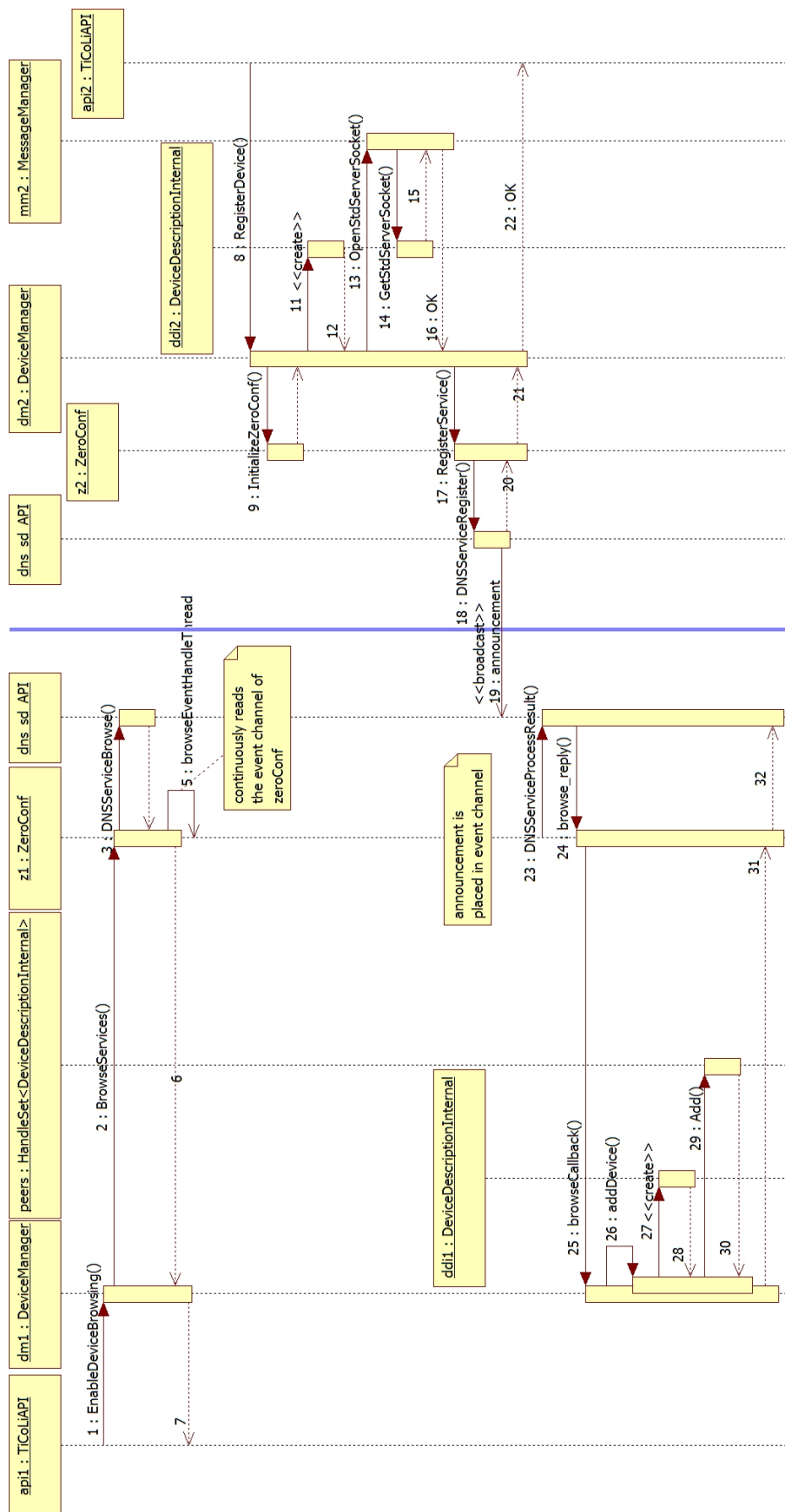


Figure 5.4: UML sequence diagram of device discovery with the TiCoLi.

- The Bonjour service sends a multicast message to all peers announcing the presence of a new device (19).
- The announcement is processed by the ZeroConf z1 service of api1 (23,24) and the DeviceManager information is forwarded to dm1 (25).
- dm1 creates a new instance ddi1 of DeviceDescriptionInternal which it adds to its mPeerDeviceList (27, 30). The name, type, and standard port of the peer device are extracted from the notification and ddi2 is parameterized accordingly.

In Figure 5.5, the DeviceManagers of three TiCoLi devices are shown. Each contains a DeviceDescriptionInternal instance mMyDevice with its own description (top row of boxes) and a description of its peers in a HandleSet (bottom row of boxes). The Handles under which an application identifies its peers are assigned locally.

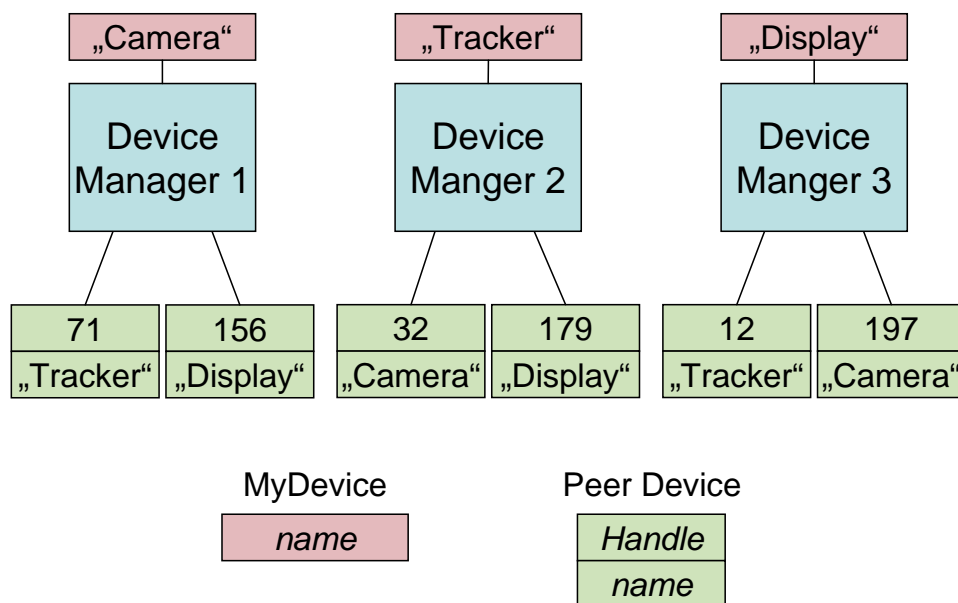


Figure 5.5: Device handles are assigned locally by the device managers of three devices.

Service Descriptions

The zeroConf protocol is only used in the TiCoLi API to announce the presence of a device in the network. The information which is included in this announcement helps to identify a device and provides the recipient with all information it requires to establish a connection with a peer. No information about the services a device is offering is contained in the announcement. To overcome this limitation, the TiCoLi contains a service discovery mechanism which complements the device discovery.

The `DeviceDescriptionInternal` instances in the `mPeerDeviceList` of a `DeviceManager` contain `HandleSets` which hold the descriptions of the services a peer offers. Separate `HandleSets` are used for the attributes to which a peer grants access, the methods which can be called on a peer, and the streams which can be obtained from it. Instances of the classes `AttributeDescription`, `MethodDescription`, and `StreamDescription` are contained in these `HandleSets`. The content of these descriptions is presented together with the services they describe in Sections 5.3.3–5.3.5.

`AttributeDescriptionInternal`, `MethodDescriptionInternal`, and `StreamDescriptionInternal` are classes which are instantiated by the `TiCoLi` managers of a device. An appropriate instance is created by the `TiCoLi` at runtime for every service an application adds to "its" `TiCoLi` instance. These instances contain all the information which is required by the `Manager` objects in the `TiCoLi` to realize the services.

While device descriptions are automatically exchanged between all registered and listening devices in the background, the exchange of the more complex service descriptions is only performed on demand. In order to obtain a service description from one server, a client calls the interface method `GetDeviceDescription(...)`. Thereby the `Handle` of a previously discovered peer device is specified. If this `Handle` is valid, the `DeviceManager` will send a request to the associated peer device. The peer's managers compile a reply message in which a description of all registered services is contained. The requesting `DeviceManager` generates service descriptions according to the content of the reply and adds them to the `DeviceDescriptionInternal` instance it holds for the requested peer.

In Figure 5.6, the device discovery mechanism is presented in an UML sequence diagram. The displayed sequence is based on the prerequisite that `app2` has already registered itself via the `RegisterDevice(...)` method and has been discovered by `app1`. Further, the scenario requires that an `AttributeDescriptionInternal` `adi` is present in the `AttributeManager` of `app2`. `adi` contains the description of an attribute of `app2` to which access shall be granted through the `TiCoLi`. Details about the `AttributeManager` and `AttributeDescriptions` can be found in Section 5.3.3.

- The client (`app1`) requests a service description of `app2` by calling `GetDeviceDescription(...)` with the `Handle` its `DeviceManager` `dm1` assigned to `ddi1`. The API forwards the call to `dm1` (1,2).
- `dm1` sends the request to the peer using the messaging service of the `MessageManager` (see Section 5.3.2) (3, 4) before it starts polling its standard client socket for the response(5).
- On the receiver side, the message is handed to the `DeviceManager` `dm2` after reception (6). `dm2` generates a response message to which it attaches information about services its application is offering (7).

- The `PackDeviceDescription(...)` call recurses through all Managers to compile a response. In the example, the `AttributeManager` is called (8), which adds the handle, name, type and read-only flag of `adi` to the response (9).
- The compiled response is returned to the sender of the request (13).
- The `MessageManager mm1` receives the message and hands it to `dm1` for handling (14, 17, 18).
- `UnpackAttributeDescriptions` recurses to all managers to have them unpack the services they are responsible for from the returned message. In the example, the `AttributeManager am1` finds an attribute described in the response and creates an `AttributeDescription ad1` according to message content. `ad1` is added to `ddi1`, the description of `app2` in the `DeviceManager` of `app1` (20 – 23).
- `GetDeviceDescription()` returns an upcasted pointer to the `DeviceDescription ddi1`.

In Figure 5.7, on the left, the `DeviceManager` and `MethodManager` of an application which controls a video camera are presented. Two methods to zoom the camera image in and out are offered by the device through the TiCoLi. On the right side of the same figure, the `DeviceManager` of an application which controls a display and obtained the service description from the first device is presented. The Handles of `MethodDescriptions` held by the `DeviceManager` of the display are identical

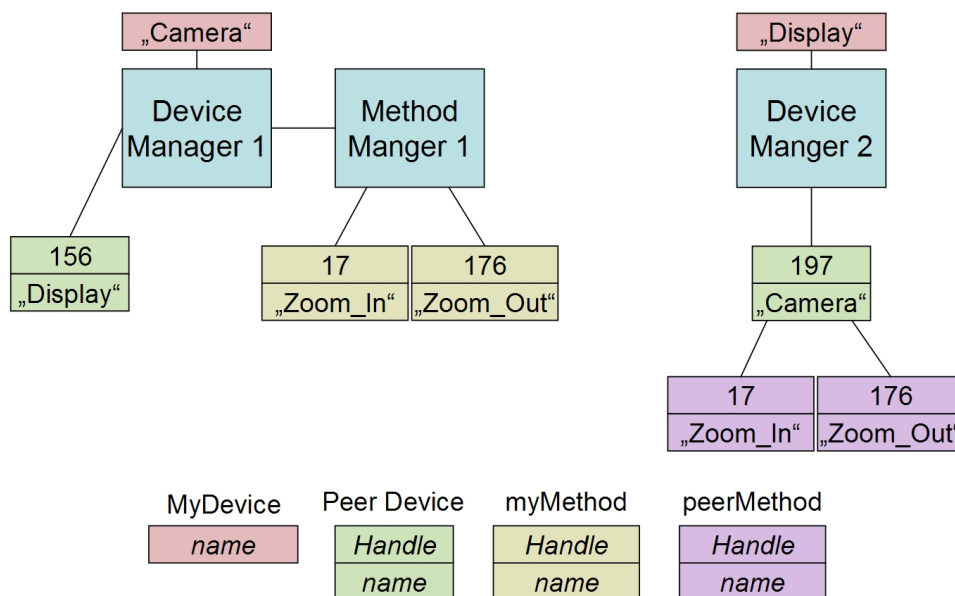


Figure 5.7: The service handles are assigned by the server. Peers identify services with the same handle as the server does.

to the `Handles` of the according `MethodDescriptionInternal` instances in the camera.

5.3.2 The Message Manager

The `MessageManager` initiates, maintains, and terminates sessions between clients and servers and is responsible for compiling, sending, and reception of messages. The `MessageManager` uses the `OpenIGTLink` library (see Appendix C) to transmit messages. The `MessageManager` and the `Message` classes are depicted in an UML class diagram in Figure 5.8.

The message service of the `TiCoLi` extends the functionalities of the `OpenIGTLink` library for message exchange as follows:

- **Acknowledged messaging:** The reception of a message is acknowledged by the recipient. This is an important feature when exchanging critical messages such as error warnings or control commands.
- **Multi-thread access:** In situations where multiple threads in an application use the messaging service at the same time, the `TiCoLi` prevents conflicting access to network resources and memory.
- **Session management with multiple peers:** An application can communicate with several peer applications through the same service without having to care about their network addresses. Peers are identified by integer handles. Applications use the handle to address messages they sent with the `TiCoLi`.

The Message classes

The `TiCoLi` messaging module contains a generic `Message` class and several subclasses for messages with specific content. The `TiCoLi` `Message` class references an instance of the `OpenIGTLink` `Message` class. In addition, the `TiCoLi` `Message` class has the following member attributes:

- `Handle mSenderHandle`: the `Handle` of the sender of a message. This parameter is assigned by the `MessageManager` of the receiver after reception.
- `MessageType mMessageType` the type of the message according to an enum definition of message types (see below). `MessageTypes` are specific to the different subclasses of `Message` and can be used to identify the correct type before downcasting a `Message` instance to the appropriate subclass.
- `mIsACK` marks messages which are sent back to the sender of a message to acknowledge message reception. With the exception of acknowledgements, all messages are acknowledged by the `MessageManager`.

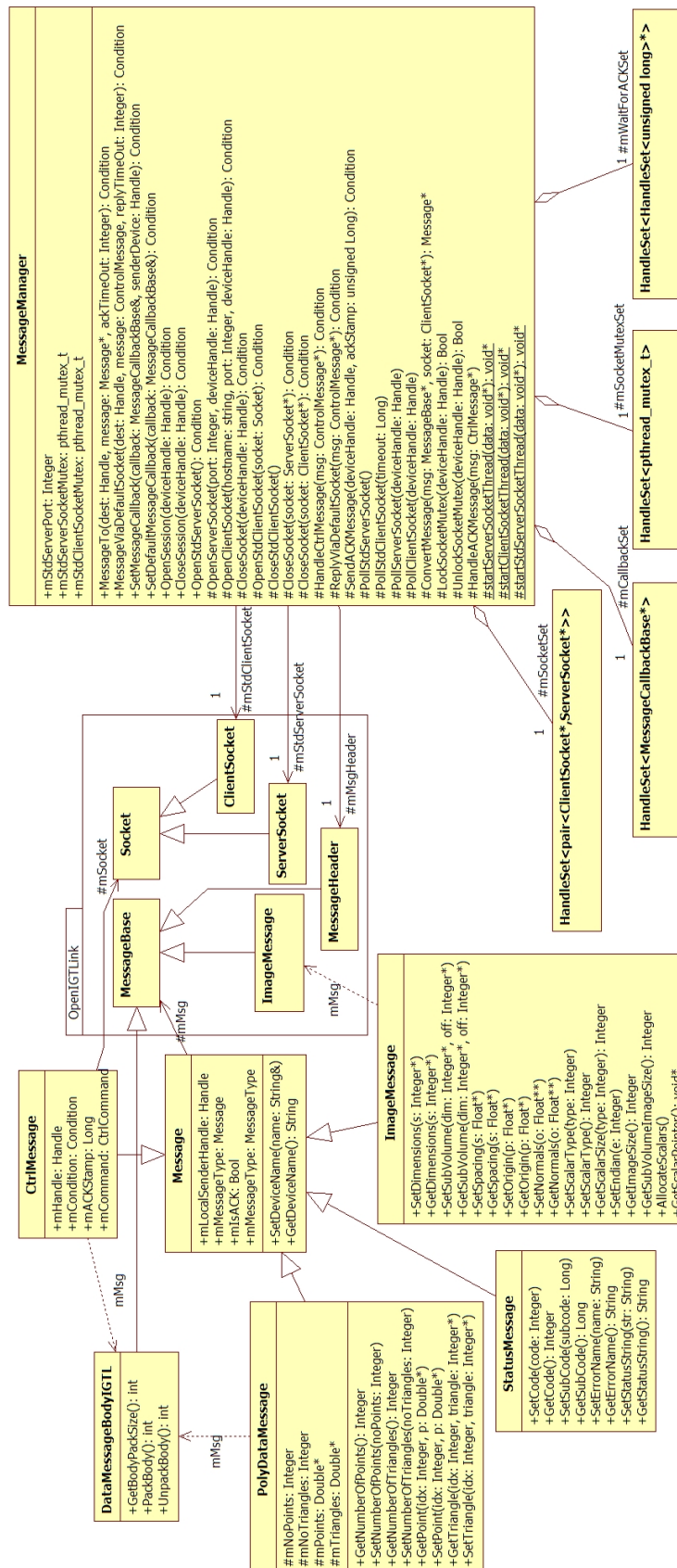


Figure 5.8: UML class diagram of the TiCoLi MessageManager and related classes.

Message Types

The following values can be assigned to the `mMessageType` of a `Message`.

- **UNKNOWN:** Default type assigned to all messages where the type is not (yet) specified.
- **STATUS:** A status message which contains two integer status codes, a status string, and an error string. The status codes are inherited from the OpenIGTLink status codes. The semantics of the status subcode is defined by the application.
- **IMAGE:** A message which contains a 2D or 3D image.
- **POLYDATA:** A message which contains a triangular mesh.
- **POSITION:** Contains a single point in 3D space.
- **TRANSFORM:** Contains a rigid affine transformation in 3D space in quaternion representation.
- **CTRL:** Special message type for control messages exchanged internally between two instances of the TiCoLi (see below).

Interface Methods

To send a message through an established session, a peer application has to create an instance of the `Message` class or an appropriate subclass and call the `MessageTo(...)` method on the `TiCoLiAPI`. A valid `Handle` which specifies the recipient of the message, a pointer to a `Message` instance and a timeout for the acknowledgement have to be set by the calling application. The `ackTimeout` is an important parameter, since the `MessageTo(...)` method will wait until the acknowledgement is received or the timeout is passed. The thread which called the `MessageTo(...)` method will not continue before either happened. The `MessageManager` does allow several threads to wait for acknowledgements simultaneously.

On the side of the message recipient, the `MessageManager` will acknowledge message reception and forward the received messages through a `messageCallback` function the application sets during initialization of the TiCoLi. `CtrlMessages` are not forwarded to the application but handed to the `TiCoLiCore::HandleControlMessage(...)` method which forwards the message to the appropriate `Manager` object (see below).

Translation to OpenIGTLink Messages

The TiCoLi `MessageManager` utilizes the OpenIGTLink library for message transport. For transfer through the OpenIGTLink service, the `TiCoLi::Message` instances are translated into `igt1::Message` instances.

Thereby, the content of a `TiCoLi::Message` which does not fit into the header specification of an `igt1::Message` (see Appendix C) is appended to the body of

the message before it is sent. Translation is encapsulated in the `Pack()` method present in the `TiCoLi::Message` class and its subclasses:

- `TiCoLi::StatusMessages` and `TiCoLi::ImageMessages` need no translation, since these classes have direct equivalents in the `OpenIGTLink` classes `igt1::StatusMessage` and `igt1::ImageMessage` which handle packing and unpacking of the message body.
- The content of `TiCoLi::CtrlMessage` and `TiCoLi::PolyDataMessage` instances is compiled into a byte stream by the sender, sent in the data block of an `OpenIGTLink` message, and reconstructed from the byte stream by the receiver.

Message transport

After the message has been compiled, the `MessageManager` selects the socket through which the connection with the receiver has been established and sends the message. To prevent several threads to send messages through the same socket at the same time, the `MessageManager` holds a semaphore for each socket. Before `ClientSocket::Send(...)` is called, the `MessageManager::MessageTo(...)` method waits for that semaphore to be free and then blocks it until it finishes sending the message. The POSIX Threads `p_mutex_t` class is used as semaphore.

The `MessageManager` maintains several `igt1::Socket` instances for network access:

- `mSocketSet` is a `HandleSet` containing socket connections with peer devices. The `MessageManager` maintains exclusive peer-to-peer session with devices in the network. Sessions are either initiated per local request through the `TiCoLiAPI` or per remote request through the network. A session consists of a `serverSocket` which is opened by the `MessageManager` on the side which initiates a session and a `clientSocket` which is opened by the `MessageManager` which accepts a connection request. The process of connection establishment is visualized in Figure 5.9.
- `mStdServerSocket` is opened when an application calls `RegisterDevice(...)`. It is held open until `UnRegisterDevice()` is called. This socket can be used by any peer device to initialize a session. During initialization of the `TiCoLi`, this socket is opened and a thread is started which continuously polls it for incoming messages. The socket is exclusively used for service discovery and message initialization. The implementation of the socket listener discards all incoming messages other than `CtrlMessages` with `mCommand == GET_SERVICE_DESCRIPTION_RQ` or `mCommand == CONNECT_MESSAGE_SOCKET_RQ`.
- `mStdClientSocket` is opened when an application establishes a connection with the `mStdServerSocket` of a peer application. This connection

is opened to either send a request for a device description or to send a request to open a message session (see above). In the first case, the connection is held open after sending for the requested device to send the reply. In the second case, the standard socket connection is closed after the request has been sent. The recipient of the request will open a messaging session with the sender and reply acknowledge through this socket.

Session Initialization

The message manager is responsible for establishing and maintaining messaging sessions. All data messages and the most `CtrlMessages` require an established session between the sender and the receiver. The only messages which can be sent without an established session are those messages which are exchanged during the handshake which is performed to identify a peer device and initialize a session with it. In Figure 5.9, the internal process of session initialization is shown:

- The application `app1` calls `OpenSession(...)` with the local `Handle` of `app2`. The API forwards the call to the `MessageManager mm1`. (1,2)
- The manager opens a server socket and starts a listening thread (3). This socket will be used for the peer-to-peer connection with `app2` if session initialization is successful.
- `mm1` locks a mutex, opens its `stdClientSocket` (5), sends a `CONNECT_MESSAGE_SOCKET_RQ` to the default port of the peer (6) and closes the `stdClientSocket` (7). After sending, `mm1` will wait for the mutex to be unlocked or a timeout of 30 seconds to pass.
- The polling thread which listens to the `stdServerSocket` of `app2` converts the incoming message (9), identifies it to be of the correct type and calls the `HandleCtrlMessage(...)` function (10).
- `app1` is added to the device list of the `DeviceManager` (11, 12) The server opens a `ClientSocket` and starts a listening thread (13). This socket is associated with the already opened socket of `app1` to build the exclusive peer-to-peer connection. All further message exchange between `app1` and `app2` will be performed through this connection.
- `mm2` sends a `CONNECT_MESSAGE_SOCKET_RSP` through the successfully established connection (14, 15) and an event is sent to `app2` (17,18,20).
- `mm1` receives the response (16, 19), checks whether it comes from a known device (21, 22) and sends an acknowledgement (23,24,25) to `mm2` (26,28,32).

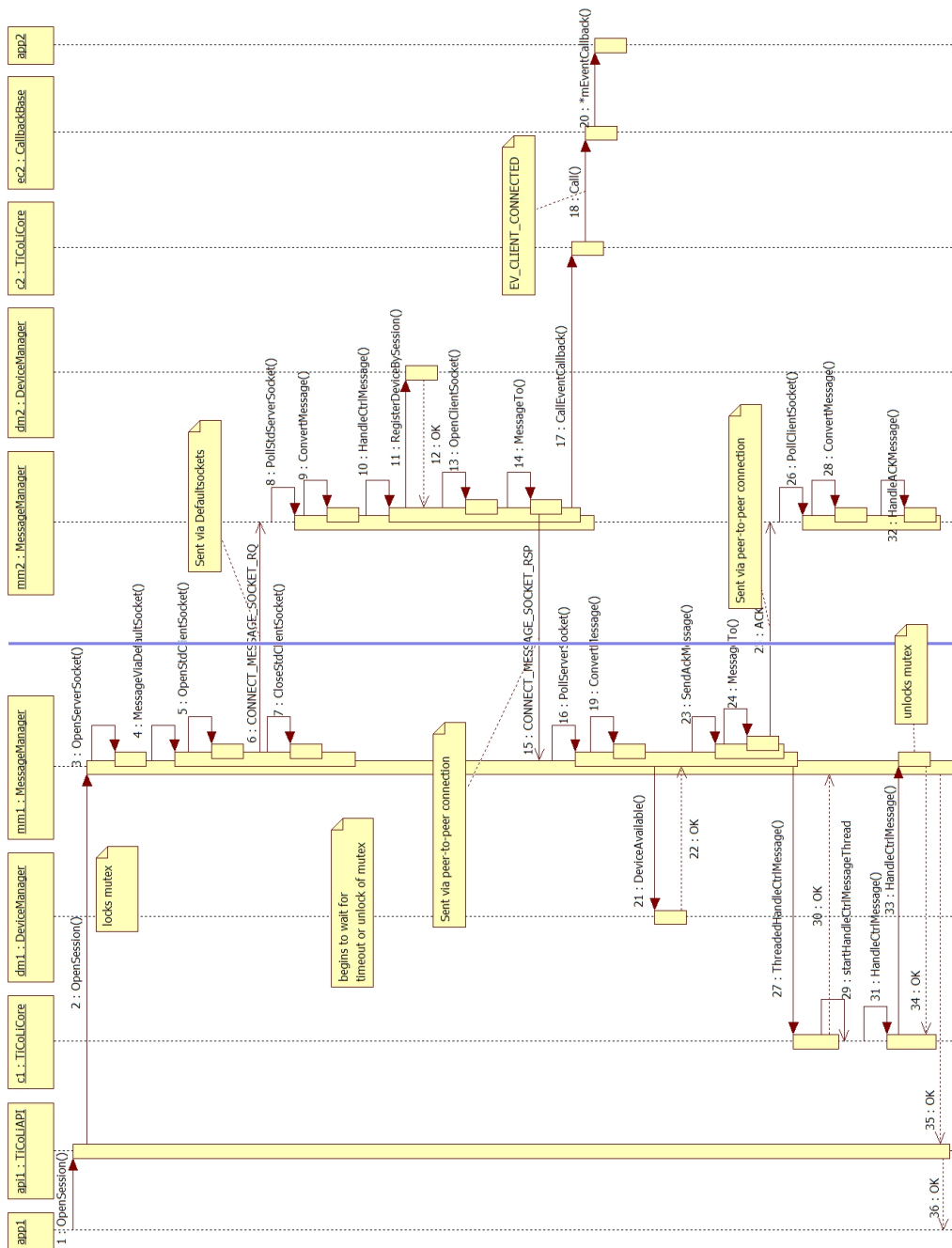


Figure 5.9: UML sequence diagram of TiCoLi session initialization.

- After sending the ACK, `mm1` initiates a thread in the `TiCoLiCore` for handling of the received `CtrlMessage`. This is the standard procedure for all `CtrlMessages` which are received via a peer-to-peer port. By handling the message in a separate thread, the `MessageManager` can go back to polling its sockets instead of waiting for the handling to be finished. In the case of an `CONNECT_MESSAGE_SOCKET_RSP`, the `TiCoLiCore` assigns the handling to the `MessageManger` which handles the event by unlocking the mutex and returning OK (27, 29, 31, 33, 34)
- The thread which was waiting for the socket to be unlocked reacts by acknowledging successful session initialization to `app1` (35, 36).

Control Messages

Control messages are exchanged to establish sessions, invoke services, and acknowledge message reception. Unlike all other message types, these messages are not public, i.e. an application cannot create them. Control messages are not handed to the application through the `messagecallback` but are processed internally by the `TiCoLiManagers`. Control messages contain a `CtrlCommand`, which identifies a specific service primitive. Depending on the nature of the command, a `CtrlMessage`, can contain additional data to parameterize the command.

The following `CtrlCommands` are defined for communication between the `DeviceManagers` of two `TiCoLi` instances:

- `UNKNOWN_CMD`: Default value assigned to control messages during creation.
- `GET_SERVICE_DESCRIPTION_RQ`: A client's requests to a server to send its complete service description.
- `GET_SERVICE_DESCRIPTION_RSP`: A server's response to the `GET_SERVICE_DESCRIPTION_RQ`. If the request could not be processed for any reason, the response contains a `Condition` which indicates the nature of the error.
- `DEVICE_MODIFIED`. When the service description of a device changes, it sends a notification to all devices it has an open session with. The notification does not contain the updated description.

The following `CtrlCommands` are defined for communication between the `MessageManagers` of two `TiCoLi` instances:

- `CONNECT_MESSAGE_SOCKET_RQ`: a client's request for a new session. The client's `TiCoLi` automatically selects an unused TCP port and includes the port number in the request.
- `CONNECT_MESSAGE_SOCKET_RSP`: a server's response to the connection request. After opening the TCP connection, this message is sent via the new connection. If the connection could not be established, the response is not sent.

- `DISCONNECT_NOTIFY`: Before an application closes a session it notifies the peer application of that session.
- `ACK`: Whenever a message is received and successfully parsed, the TiCoLi automatically sends an acknowledgement. Note: Only message reception is acknowledged. An `ACK` neither implies that the TiCoLi or the application to which the message is forwarded could make any sense of the message nor that the recipient reacts to it in any way.

Additional control messages are defined for requesting stream connections, access to attributes, calling of methods and the responses to these requests. These messages will be described together with the services in the sections below. The `MessageManager` hands `CtrlMessages` to the `HandleCtrlMessage(...)` method of the `TiCoLiCore` from where they are forwarded to the appropriate manager, depending on the `CtrlCommand` (see Figure 5.10).

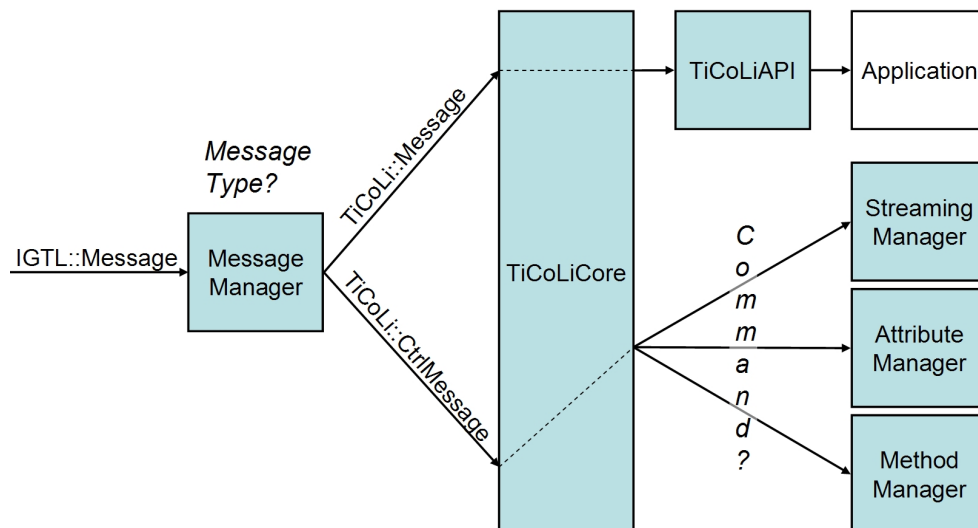


Figure 5.10: Message distribution inside the TiCoLi.

5.3.3 The Attribute Manager

The `AttributeManager` class implements all methods required to exchange attribute values between two TiCoLi applications. In Figure 5.11, the `AttributeManager` class and related classes are depicted.

Attribute Service Descriptions

The `AttributeManager` aggregates instances of the `AttributeDescriptionInternal` class. Each of these instances contains the properties of one attribute the application has added to the manager for publication in the network and

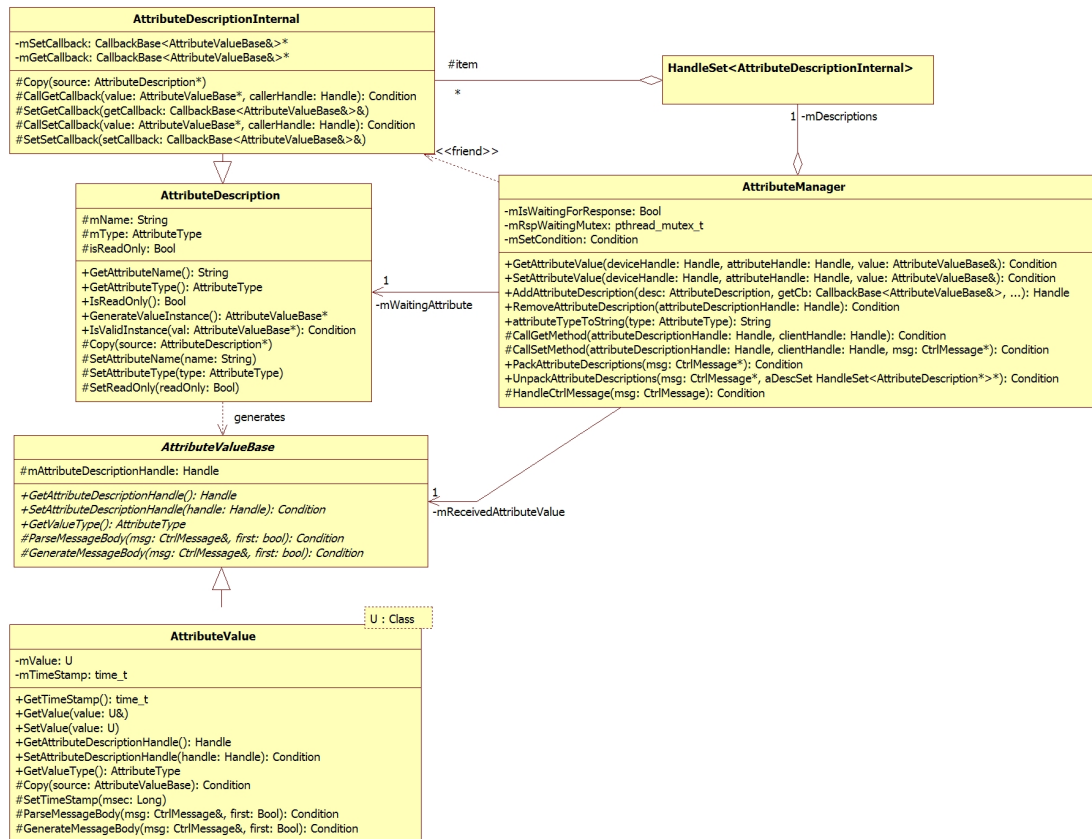


Figure 5.11: UML class diagram of the TiCoLi AttributeManager and related classes.

callback pointers to set- and get-methods for the attribute. The instances of this class are only held internally by the AttributeManager of a server.

The AttributeDescription is a superclass of AttributeDescriptionInternal. It does not contain the callback pointers. AttributeDescription instances are used in the AttributeManager object of a TiCoLi instance to represent the attribute services of peer devices. AttributeDescriptions are exchanged during service discovery. The class contains the following member variables:

- `string mName`: The name of the attribute. The name is set by the application which shares the attribute.
- `AttributeType mType`: The type of the attribute. The type is selected from an enum type definition (see below).
- `bool mIsReadOnly` a Boolean which distinguishes read-only from read-write attributes.

Attribute Types

The TiCoLi attribute service offers the following attribute types:

- `AT_UNKNOWN`: Standard type assigned during construction and to attribute descriptions which could not be parsed correctly.
- `AT_BOOL`: Boolean attributes with values `true` and `false`.
- `AT_INT`: signed 32 bit Integers between -2^{31} and $2^{31} - 1$.
- `AT_SHORT`: signed 16 bit Integers between -2^{15} and $2^{15} - 1$.
- `AT_LONG`: signed 64 bit Integers between -2^{63} and $2^{63} - 1$.
- `AT_UNSIGNED_INT`: unsigned 32 bit Integers between 0 and $2^{32} - 1$.
- `AT_UNSIGNED_SHORT`: unsigned 16 bit Integers between 0 and $2^{16} - 1$.
- `AT_UNSIGNED_LONG`: unsigned 64 bit Integers between 0 and $2^{64} - 1$.
- `AT_DOUBLE`: double-precision floating point numbers according to the IEEE-754 standard [IEEE, 1985].
- `AT_STRING`: character string of arbitrary length.

The attribute service does not contain a mechanism for constructing complex data types by composing these types in a record-like structure.

AttributeValue Classes

The `AttributeValueBase` class and its subclasses hide the implementation details of attribute value exchange from the application. `AttributeValueBase` is an abstract interface which contains declarations of all type-unspecific functionalities. `AttributeValue<class U>` is the templated definition of an attribute value of any of the types the TiCoLi supports (see above). These classes are used at the interface between an application and the TiCoLi API to exchange attribute values.

To indicate to which `AttributeDescription` an `AttributeValue` belongs, `AttributeValueBase` contains the `Handle mAttributeDescriptionHandle` parameter.

The `GenerateValueInstance()` of `AttributeDescription` can be used to generate an `AttributeValue` (see below). `Condition IsValidInstance(...)` can be called on an `AttributeDescription` to check whether an `AttributeValue` belongs to an `AttributeDescription`.

The classes contain protected methods for compilation and parsing of messages which are utilized during `CtrlMessage` packing and unpacking and templated `Get-` and `SetValue` methods for the application to conveniently access the encapsulated attribute value.

Interface Methods

To publish an attribute value, a server has to create an `AttributeDescription` instance and by calling the constructor `AttributeDescription(...)` and then

call the method `Handle AddAttributeDescription(...)`. Callback functions for read- and write-access to the attribute have to be specified by the application. No `setCallback` is required for read-only Attributes. The method returns the `Handle` which the `AttributeManager` assigned to the added description. The application can use this handle to withdraw the description at a later point in time by calling `Condition RemoveAttributeDescription(...)`. The `Handle` is part of the `ServiceDescription` which is sent to peer devices. Peers use the `Handle` to refer to a specific attribute when requesting access to it.

To access an attribute in a server, a client has to create an appropriate container for the value by calling `GenerateValueInstance()` on the `AttributeDescription` of that attribute. The method is templated so that it creates instances of the correct subtype. For example, `GenerateValueInstance()` will generate an instance of `AttributeValue<double>` when called on an `AttributeDescription` with `type == AT_DOUBLE`. The client can then access the attribute by calling `GetAttributeValue(...)` or `SetAttributeValue(...)`, respectively on its `TiCoLiAPI`. The API will forward this call to its `AttributeManager` which will send a request to the server and wait for the response.

Control Messages

The following `CtrlCommands` are defined for communication between the `DeviceManagers` of two `TiCoLi` instances:

- `GET_ATTRIBUTE_RQ`: A client's request for the actual value of an attribute. The request contains the server-side `Handle` of the attribute.
- `GET_ATTRIBUTE_RSP`: A server's response to an `GET_ATTRIBUTE_RQ`. If possible, the response contains the attribute value. A `Condition` is contained which is `OK` if attribute access was successful and an appropriate error code otherwise.
- `SET_ATTRIBUTE_RQ`: A client's request for the actual value of an attribute. The request contains the server-side `Handle` of the attribute and the value the client wants to set.
- `SET_ATTRIBUTE_RSP`: A server's response to a `SET_ATTRIBUTE_RQ`. A `Condition` is contained which is `OK` if attribute access was successful and an appropriate error code otherwise.

Example

The UML sequence in Figure 5.12 diagram depicts the internal processing of a `SetAttribute(...)` call on the `TiCoLi API`. For the sakes of space and clarity, the session initialization and service discovery phases are omitted in the diagram.

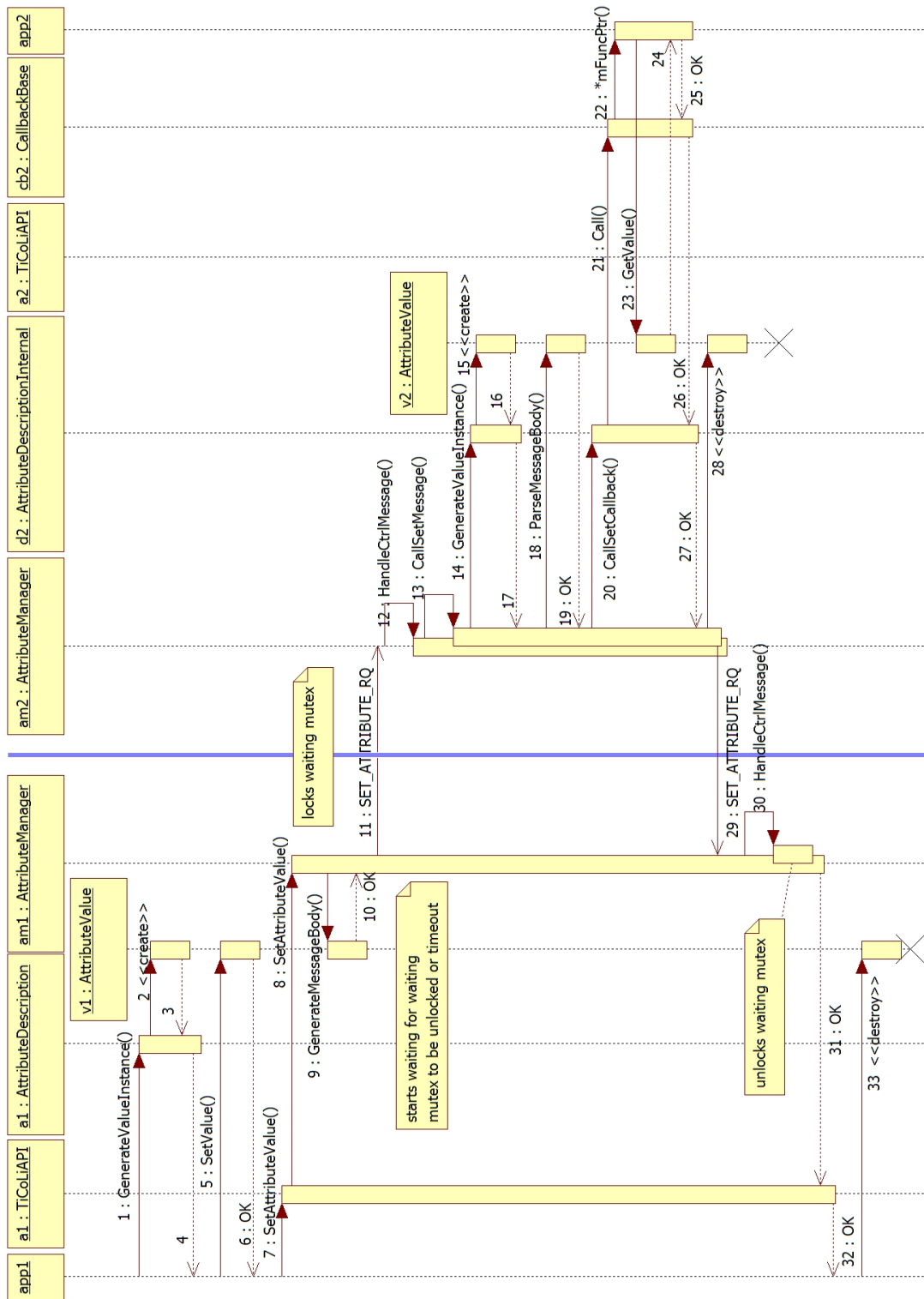


Figure 5.12: UML sequence diagram of TiCoLi attribute access.

- `app1` generates an instance `v1` of `AttributeValue` by calling `GenerateValueInstance()` on its `AttributeDescription d2` and assigns a value to that instance (1 – 6).
- The API call `SetAttributeValue(...)` is forwarded to the `AttributeManager am1` which creates a `CtrlMessage` with the `SET_ATTRIBUTE_RQ` command. The message is handed to `v1` which packs its content into the body before the message is sent to the peer (7 – 11). `am1` locks a mutex and goes to a busy waiting state until the mutex is unlocked or a timeout is reached.
- Sending, reception, and acknowledgement of the message with the `MessageManagers` of both `TiCoLi` instances are omitted in the diagram.
- Handling of the received `CtrlMessage` is delegated to `am2` (12, 13), which calls `GenerateValueInstance()` on `d2` to generate `v2` (14 – 17). `v2` extracts the sent attribute value from the message body (18, 19). The request to set the attribute is handed through to `app2` via the `mSetCallback` instance associated with `d2` (20 – 22). `app2` gets the designated attribute value from `v2` (23) and returns OK. `v2` is deleted afterwards (28).
- The returned `Condition`, (OK in the example) is put as an argument into the `SET_ATTRIBUTE_RSP` message which is sent to `am1` (29).
- The `MessageManager` starts a thread in which it calls `HandleCtrlMessage()` on `am1`. The response is parsed and the mutex is unlocked (30). The next time the thread in which `SetAttributeHandling(...)` is running checks the mutex, it will registers the successful execution of the request and returns OK (31,32).
- `app1` deletes `v1` (33).

5.3.4 The Method Manager

The `MethodManager` class implements all functions required to exchange method calls and return values between two `TiCoLi` applications. Figure 5.13 shows the `MethodManager` and other classes related to the remote method service.

Method Service Descriptions

The method manager aggregates instances of the `MethodDescriptionInternal` class. Each of these instances represents one method the application grants access to via the `TiCoLi`. Only abbreviated versions of these descriptions are sent to peer applications as part of the `GET_SERVICE_DESCRIPTION_RSP` messages during service discovery. The `MethodDescription` which is sent to the peers and handed out to the applications "behind" the peer `TiCoLi` instances contain for each method:

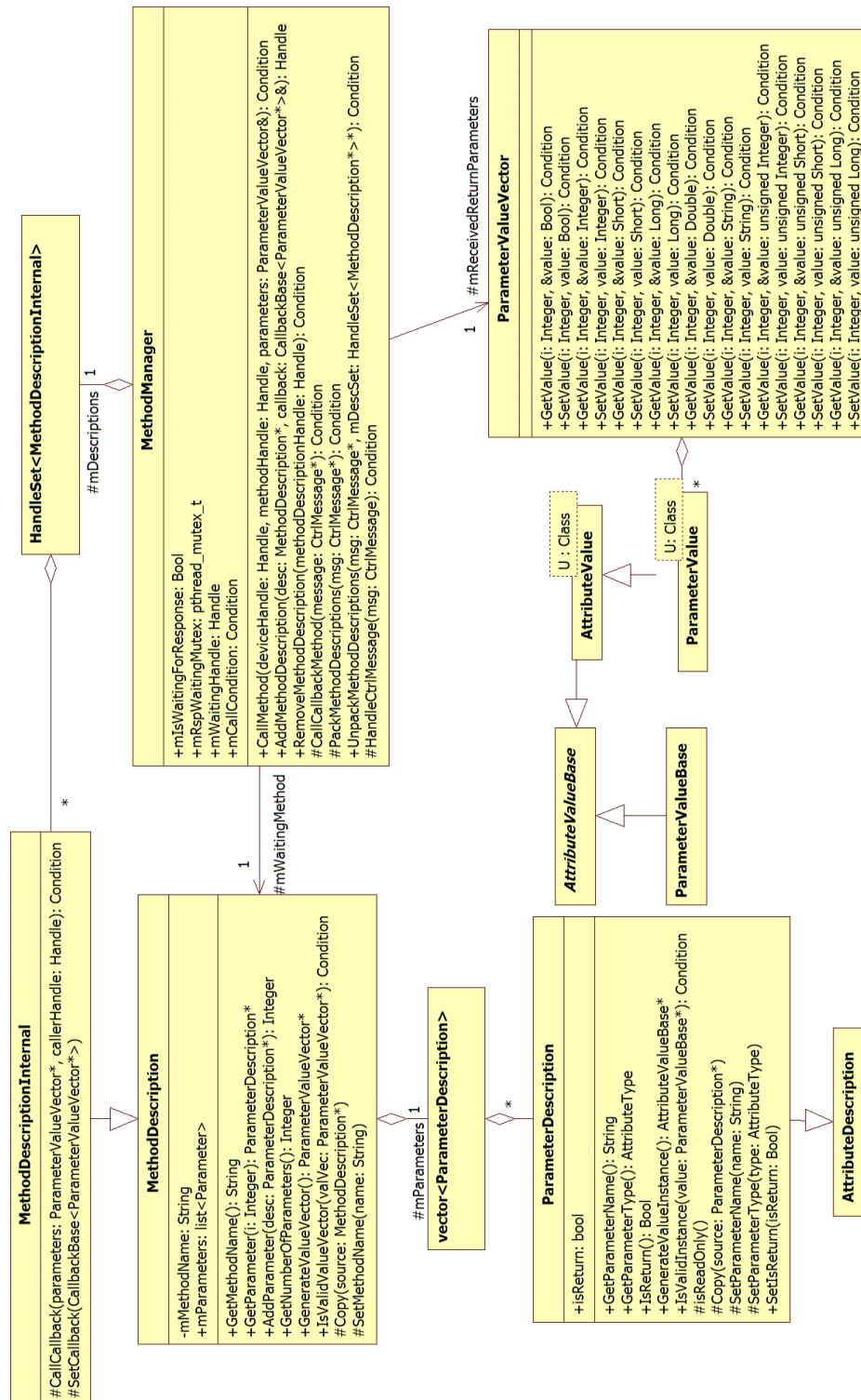


Figure 5.13: UML class diagram of the TiCoLi MethodManager and related classes.

- `string mName`: The name of the method. The name can be freely chosen by the application.
- `vector<ParameterDescription*> mParameters`: A vector containing the declaration of the input and output parameters of the method (see below).
- `mEstimatedRuntime`: An estimate of the number of seconds the caller of a method has to wait for the result.

In addition to these parameters, the `MethodDescriptionInternal` held by a server's `MethodManager` contains a function pointer to the method of application which it shall call when a client requests to call the method. The signature of this method is required to be `(ParameterValueVector*, Handle)`.

Parameter Descriptions and the ParameterValueVector

The `ParameterDescription` class is a subclass of the `AttributeDescription` class. It is used to specify the input and output parameters of a method. The declaration of `ParameterDescription` adds the attribute `bool mIsReturn` to the `AttributeDescription` which distinguishes between input and output parameters. Otherwise, the `ParameterDescription` class behaves exactly as the `AttributeDescription` class does.

As the `AttributeValue` classes provide an interface to the value of an attribute, the `ParameterValue` classes provide an interface to the value of a parameter. The specification and implementation of these classes is identical apart from the different names.

The `ParameterValueVector` class is a container class which aggregates all parameters of one method. The container is implemented as a vector, i.e. as a linked list of dynamic length which allows indexed access to the contained items. The `MethodDescription` class contains a method which generates a `ParameterValueVector` of correct length and with the correct parameter types for the method it describes. This class is used to exchange parameter values between the TiCoLi API and the applications which calls a method through the TiCoLi or which contains the called method.

Interface Methods

To make a method available for other devices in the network, a server creates and parameterizes a `MethodDescription` and adds the required input and output parameters to it. The interface method `AddMethodDescription(...)` method of the TiCoLi API is used to add the description to the `MethodManager`. Thereby, a callback pointer to a function of the application has to be specified.

To call a method in a server, a client has to create an appropriate `ParameterValueVector` object by calling `GenerateValueVector()` on the `MethodDescription` of the method it wishes to call. After filling in all the input values

in the `ParameterValueVector`, the client calls `CallMethod(...)` on the TiCoLi API.

When the `MethodManager` of a server receives a valid request to execute a method, it unpacks the contained `ParameterValueVector` and calls the callback function with that vector. The application has to read all input parameters from the vector, execute the method and fill the results into the `ParameterValueVector` which is packed into the response message (see below) by the `MethodManager`.

Control Messages

The following `CtrlCommands` are defined for communication between the `MethodManager`s of two TiCoLi instances:

- `CALL_METHOD_RQ`: a client sends a request to execute a method to a server. The message contains the handle of that method and an appropriate `ParameterValueVector`.
- `CALL_METHOD_RSP`: after executing a requested method, the server sends a response containing the status of the execution and, if executed successful, a `ParameterValueVector` with updated return parameters.

Example

The UML sequence diagram in Figure 5.14 depicts the internal processing of a `CallMethod(...)` call on the TiCoLi API. In the example, a method is called which has one input and one output parameter. For the sakes of space and clarity, the session initialization and service discovery phases are omitted in the diagram.

- `appl` generates a `ParameterValueVector` by calling `GenerateValueVector()` on the `MethodDescription` `md`. Value instances for both the input and the output parameters are generated and returned in a `std::vector` container (1 – 8). The application assigns a value to the input parameter `p1` (9, 10).
- The API call `CallMethod(...)` is forwarded to the `MethodManager` `m1` which creates a `CtrlMessage` with the `CALL_METHOD_RQ` command. `p1` is attached to the body of that message before it is sent to the peer (13 – 15). `mm1` locks a mutex and goes to a busy waiting state until the mutex is unlocked or a timeout is reached.
- Sending, reception, and acknowledgement of the message in the `MessageManagers` of both applications are omitted in the diagram.
- Handling of the received `CtrlMessage` is delegated to `mm2` (16, 17), which generates a `ParameterValueVector` by calling `GenerateParameterVector` on `mdi` (18 – 25) The value of `p3`, the input parameter, is extracted

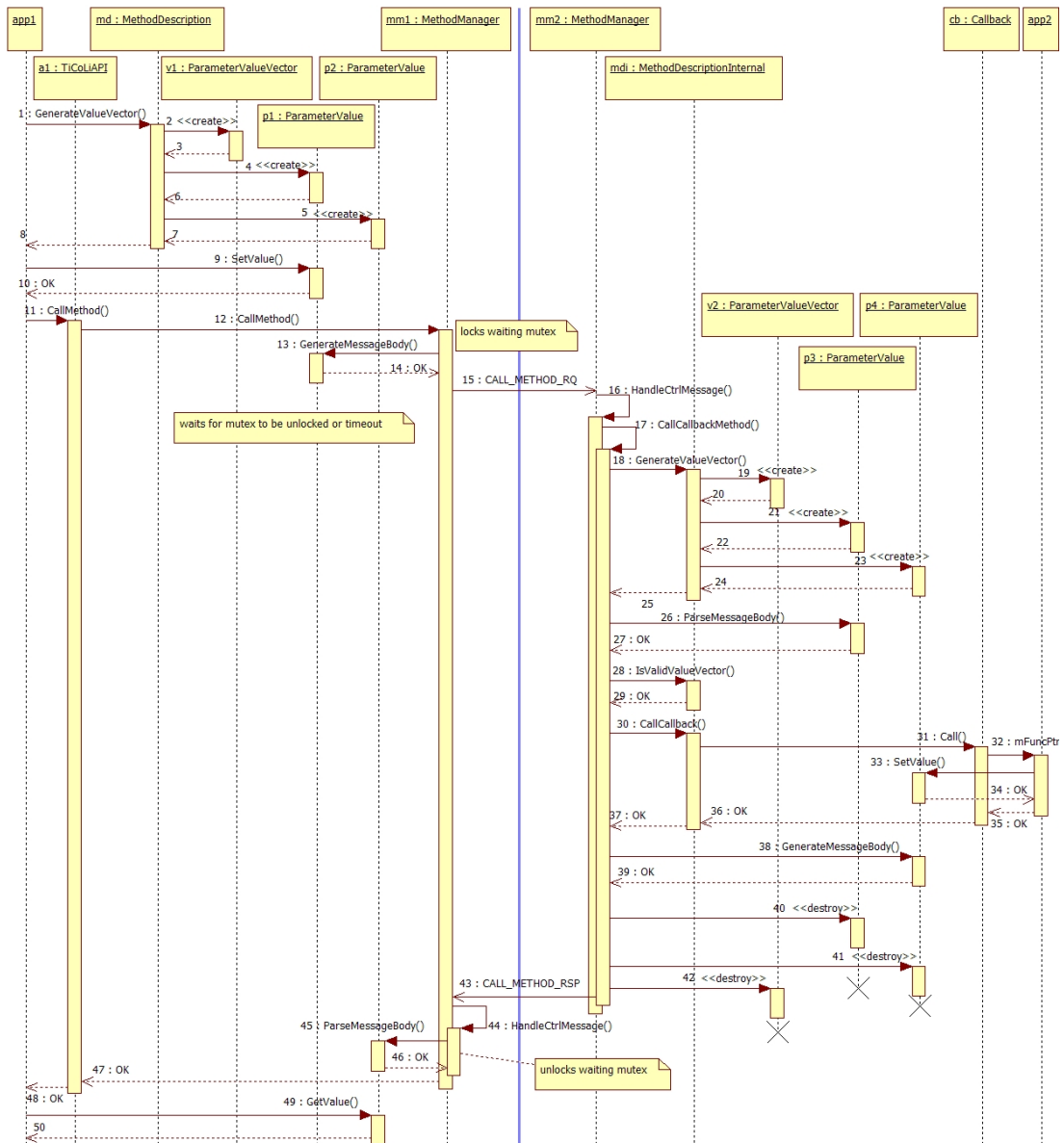


Figure 5.14: UML sequence diagram of a TiCoLi remote method call.

from the request (26, 27). The validity of the generated and parameterized vector is approved (28, 29) and the callback is executed with *v2* as argument (31 – 32). After executing the method, the application *app2* assigns its result to *p4* (33, 34) before returning OK (35 – 37).

- *m2* prepares a `CALL_METHOD_RSP` message to which's body *p4* is added (38, 39) before *v2* is deleted (40 – 42) and the message is sent back to *app1* (43).
- The incoming `CtrlMessage` is delegated to *mm1* (44). The returned `ParameterValue` is unpacked (45, 46) and the mutex is unlocked.

- The waiting thread finds the mutex unlocked and acknowledges successful method execution and result reception to `app1` (47, 48).
- `app1` gets the result value from `p2` (49, 50).

5.3.5 The Streaming Manager

The `StreamingManager` of the `TiCoLi` holds all information about the streams an application is offering or actually broadcasting as well as of the streams it is receiving from peer applications. The member attributes and functions of the `StreamingManager` and its associated classes are shown in Figure 5.15. The `StreamingManager` holds a `HandleSet<StreamDescriptionInternal>`. In this container, the information about the streams the local device is offering through the `TiCoLi` is organized. Additional `HandleSets` organize the outbound and inbound streams the manager is currently sending or receiving.

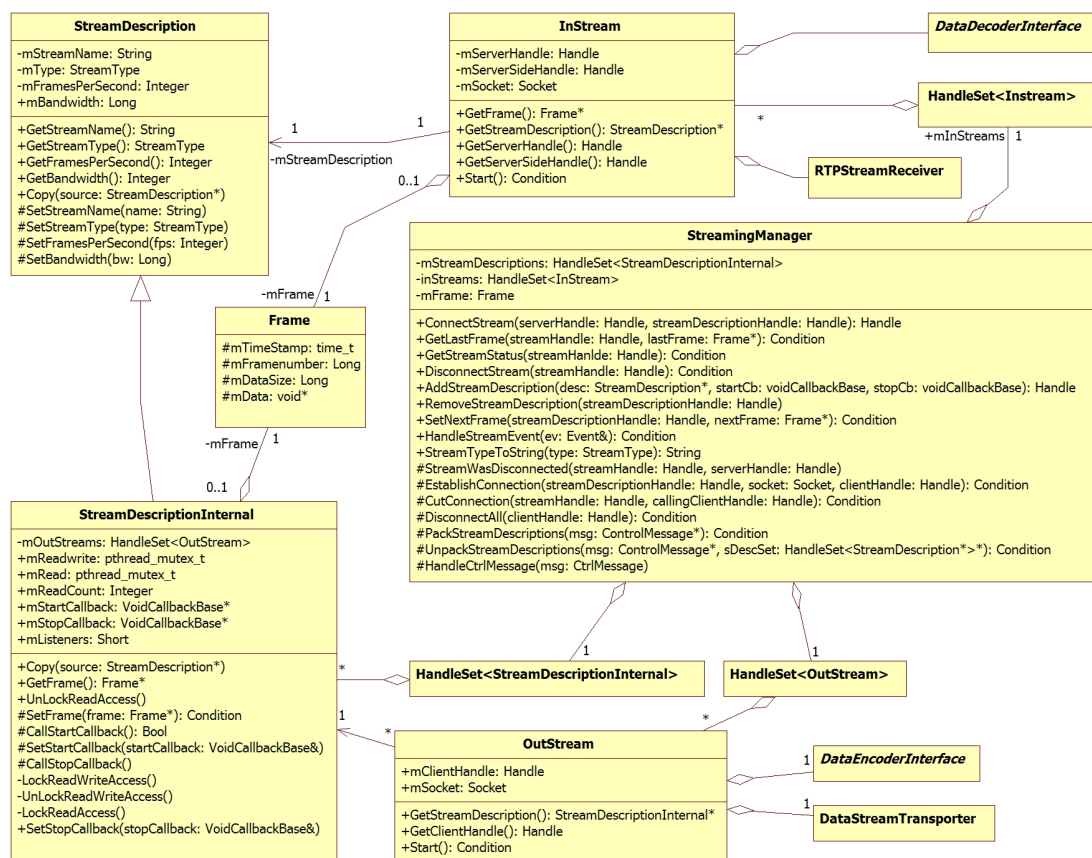


Figure 5.15: UML class diagram of the TiCoLi StreamingManager and related classes.

Stream Descriptions

The `StreamDescription` class contains all information a device sends to a peer device to describe a stream it is able to send. `StreamDescriptions` are exchanged between the sender and receiver of a stream and are the service level agreement on which the streaming connections are founded. A `StreamDescription` contains the name of a stream, its `StreamType` (see below), framerate and required bandwidth for transmission. `StreamDescriptions` are exchanged through the device discovery service implemented in the `DeviceManager`.

The `StreamDescriptionInternal` class contains all information a device stores internally about a stream it is able to send to one or more recipients. `StreamDescriptionInternal` inherits from `StreamDescription` and adds the following class members:

- A callback pointer to a functions in the application which the `StreamingManager` calls when the first client connects to a stream.
- A callback pointer to a functions in the application which the `StreamingManager` calls when the last client disconnects a stream.
- A number of mutexes and counters required to regulate read- and write access to the frames in a stream.
- The next frame which is to be sent by all outbound streams which refer to a `StreamDescriptionInternal`.

Instreams and Outstreams

The `InStream` class is created by the `StreamingManager` of an application which receives a stream. An `InStream` aggregates decoders and interface classes for reception of RTP streams. An `InStream` has a `Handle` and aggregates a `Frame` instance into which the decoder stores the last decoded frame. To facilitate the communication between peer `MessageManagers`, an `InStream` contains the `Handle` of the associated `OutStream` on the server side.

Instances of the `OutStream` class are created and managed by a `StreamDescriptionInternal`. Several `OutStreams` can be instantiated of the same `StreamDescriptionInternal` in order to allow multiple clients to listen to a stream. `OutStreams` aggregate encoders and interface classes for sending RTP streams. An `OutStream` has a `Handle`. The `Frame` of outbound streams is not stored in the `OutStream` instances but in the `StreamDescriptionInternal` instance to which an `OutStream` refers. This saves memory and memcpy operations. To facilitate peer-to-peer communication between `StreamingManagers`, an `OutStream` contains the `Handle` of the receiver's `InStream`.

In Figure 5.16, a scenario is presented where two devices receive video streams from an application which controls a video camera. Both receivers' `StreamingManagers` created an `InStream` to receive the streams. The `StreamDescriptionInternal` on the server side created and manages the according `OutStreams`.

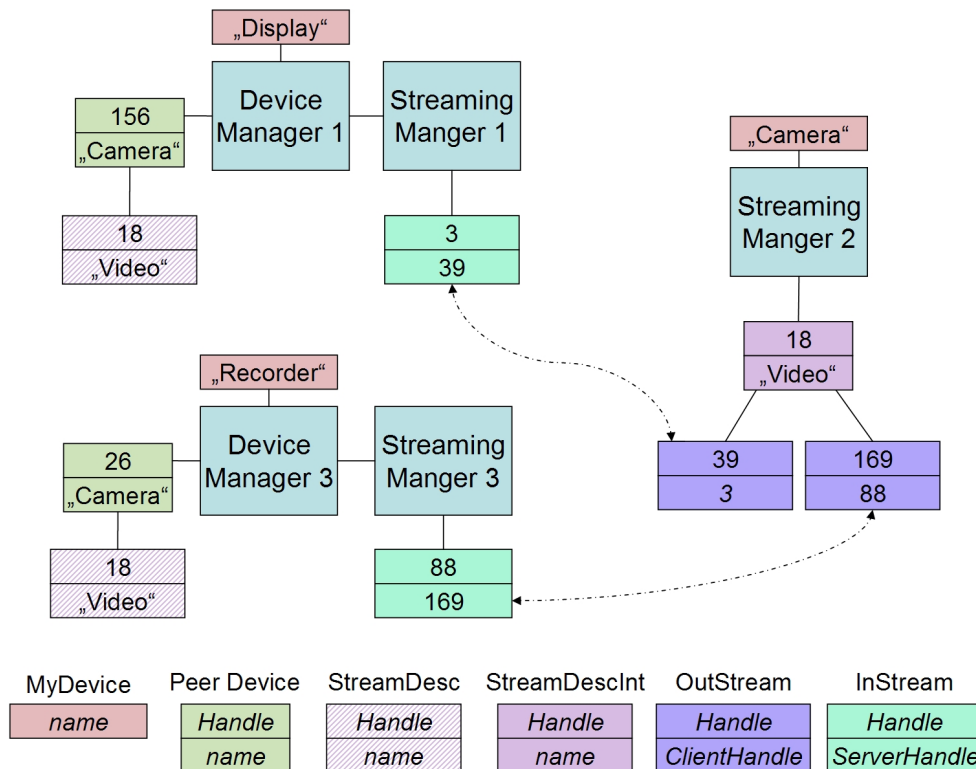


Figure 5.16: Two devices, a display and a recorded, receive a video stream from a camera.

Encoding and Decoding of RTP Streams

Different encoders are required for different kinds of streamed data. Video data, e.g., requires compression in an mpeg or other video encoder, whereas sensor data will be transmitted uncompressed and therefore without any added noise. The UML class diagram in Figure 5.17 shows the relations between frames, encoders, decoders, and sessions. The design and implementation of the encoders and their interaction with the jRTP library are based on the works of Arun Voruganti from ICCAS in Leipzig, Germany.

Interface Methods

To add a stream to its services, a server application has to create a StreamDescription instance and parameterize it accordingly. By calling `Handle AddStreamDescription(...)`, the stream is added to the StreamingManager. The function returns the `Handle` which the StreamingManager assigned to the added description. The server application can use this handle to withdraw the description at a later point in time by calling `Condition RemoveStreamDescription(...)`. All connections with clients will be cut if a `StreamDescription` is withdrawn.

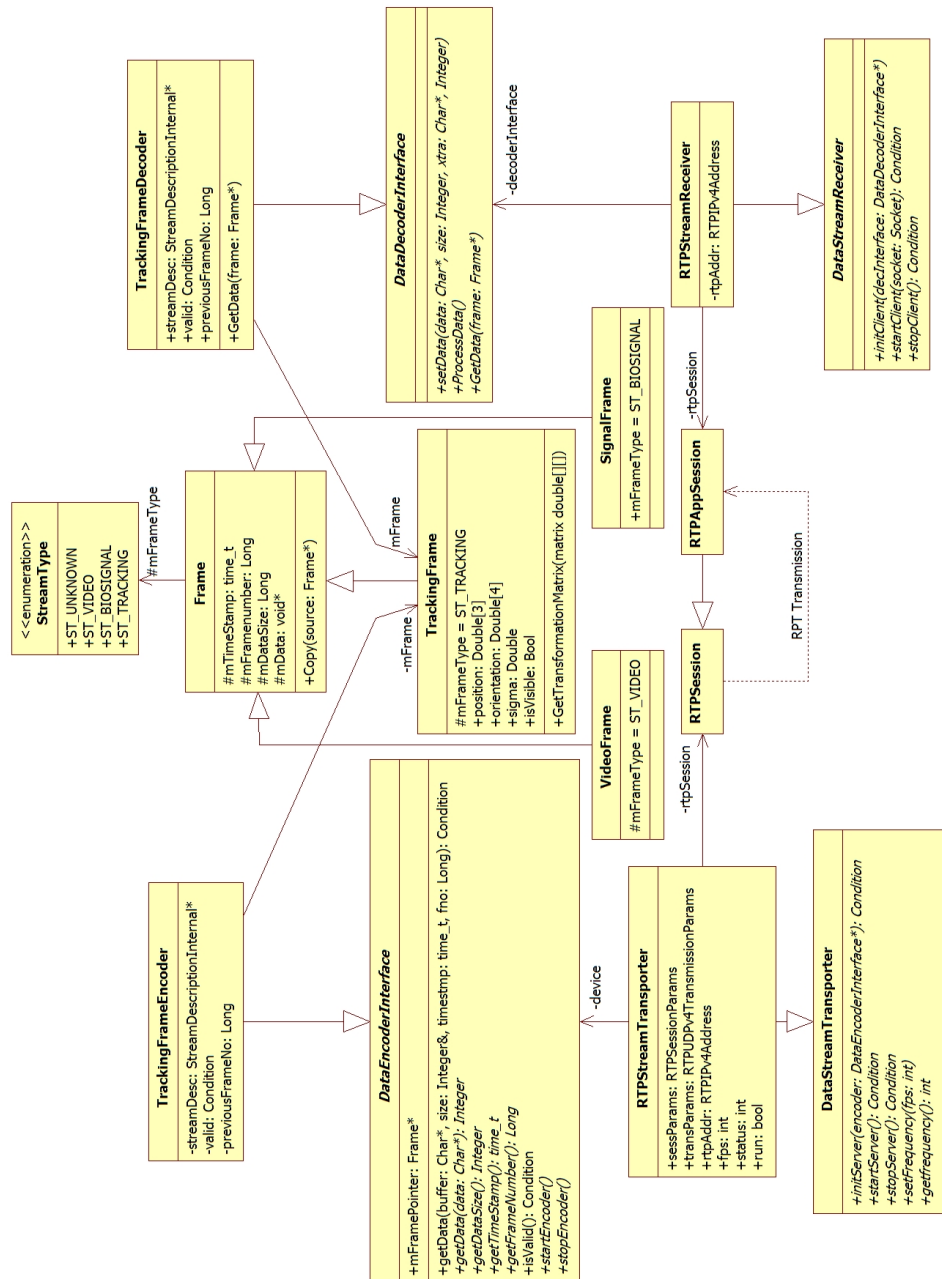


Figure 5.17: UML class diagram of the TiCoLi frame and related classes.

Once a stream is started, the server application has to feed data into the stream by calling `SetNextFrame(...)`. To connect to a stream, a client calls `Handle ConnectStream(...)`. The `Handle` of the new `InStream` is returned. With this handle, the client application can refer to the stream for access to received frames by calling `Condition GetLastFrame(...)` or to terminate the connection by calling `DisconnectStream(...)`.

Neither client nor server needs to specify or maintain network sockets for the connection. Selection of free ports and establishment of the RTP connection are hidden behind the TiCoLi API.

Stream Synchronization

The streaming service of the TiCoLi takes the task of synchronizing the generation of frames with the sending frequency of the stream from the application. All `OutStreams` run by a server will send frames to their recipients at exactly the frame rate which is specified in the `StreamDescriptions` of these streams, regardless of the rate at which the application assigns new frame data. If a server application sets frames faster than they are actually sent by an `OutStream`, some frames will be skipped. If a server application falls behind the pace at which an `OutStream` sends frames, the last frame will be sent again and again until the application provides new data. The recipient of a stream is able to detect both skipped frames and double-posted frames by comparing the frame numbers of the incoming frames.

On the recipient's side, the TiCoLi API de-synchronizes the stream. Internally, threads are running which grab and process every frame in time (provided that there are sufficient hardware resources to do so) and make it available in the `mFrame` object of the `InStream`. The client application can access that frame in an asynchronous manner by calling the `GetLastFrame(...)` function on the API which will return `NO_NEW_DATA` if no frames were received since the last call.

Control Messages

The following `CtrlCommands` are defined for communication between the `StreamingManagers` of two TiCoLi instances:

- `CONNECT_STREAM_RQ`: A client requests a new streaming connection with a server. The message contains the socket which the client has opened to receive the stream.
- `CONNECT_STREAM_RSP`: A server reports on processing of a `CONNECT_STREAM_RQ` to the requesting peer. If the stream could successfully be initialized, the message contains the socket from which the server starts sending the frames. Otherwise, an error code is contained.
- `DISCONNECT_STREAM`: A client with an open streaming connection informs a server that it will no longer be listening.

Manager `s1` which obtains a pointer to `sd` from the `DeviceManager` (2 – 4), composes an appropriate `CtrlMessage` with the command `CONNECT_STREAM_RQ`, and sends it to the peer (5). `s1` will wait up to 30 seconds the mutex to be unlocked.

- On the server side, the received `CtrlMessage` is forwarded to the `StreamingManager sm2` (6,7). An `OutputStream os` is created (8, 9) and added to the `StreamDescriptionInternal sdi`'s outstreams. When started, `os` retrieves all relevant parameters from `sdi` (10 – 15) and initializes an RTP session.
- The `startCallback` of `sdi` is called. The application starts to generate frames and assign them to the outbound streams (16 – 22).
- A `CONNECT_STREAM_RSP` is sent to the client (23). The response contains the network address of the socket from which the server will send the stream.
- The `MethodManager` of the client receives the response and creates an `InStream` instance according to the `StreamDescription sd`.
- The streaming connection is established and the `OutputStream` begins to send frames at a rate according to the value of `mFrameRate` in `sdi` (41, 54, 65, 76).
- The server application calls `SetNextFrame` on `sdi` to assign new frame data to the stream. The server application does not have to set frames at the exact transmission rate. The encoder classes handle situations where frames are set too often or not often enough. As an example, in step 54, the same frame is sent by the encoder which had already been sent in step 41, because the application did not assign new data in the meantime.
- The client application calls `GetLastFrame(...)` for pull-access to an `InStream`. The incoming frames are not buffered, i.e. only the very last received frame can be obtained by `appl`. `GetLastFrame(...)` returns `NO_NEW_DATA` when no new frame was received since the last call (see steps 52 and 62).

5.4 Performance Tests

The TiCoLi API was tested with respect to the performance of the messaging and the streaming services. The following success criteria were defined for the tests:

1. Status messages are delivered instantaneously, i.e. within less than 20 *ms* regardless of the network load.
2. Given sufficient network bandwidth, the streaming service delivers frames at the intended frame rate,

3. The TiCoLi does not add latency to frame and message delivery more than 20 *ms*. For instance, delivery of a 10 *MB* message may not take longer than 8020 *ms* over a 10 *Mbit* network.

5.4.1 Streaming Throughput and Reliability

In a first series of experiments the reliability of the streaming service was tested. For these tests, a server application was implemented which sends a stream at an adjustable framerate and frame size. A client was implemented which continuously queries the TiCoLiAPI for new frames and measures the delay between frames with a temporal resolution of 1 *ms*. A log file is created which contains for every frame the time which passed between reception of this and the previous frame. After a test period of 60 seconds, the log is stored and the mean and variance of the frame rate during the test period can be computed from the log entries. The experiment was conducted with two different setups and a variety of frame rates and frame sizes. Beginning with a 1 *Hz* stream of 1 *kByte* frames, both parameters were independently increased up to 64 *Hz* and 64 *kByte*, resulting in a maximum bandwidth of 32 *Mbit/s* for one stream.

In the first series of experiments, four servers and four clients were connected via a dedicated fully switched 1 *Gbit/s* network (see Figure 5.19, right). The servers were equipped with 100 *Mbit/s* network interface cards, in the clients, 1 *Gbit/s* network interface cards were used. The nominal frame rates and frame sizes of all four servers were kept identical in all trials. The mean and variance of the effective framerate measured all four clients were recorded according to the description above with 81 different stream parameters with bandwidths ranging from 8 *kbit/s* per stream

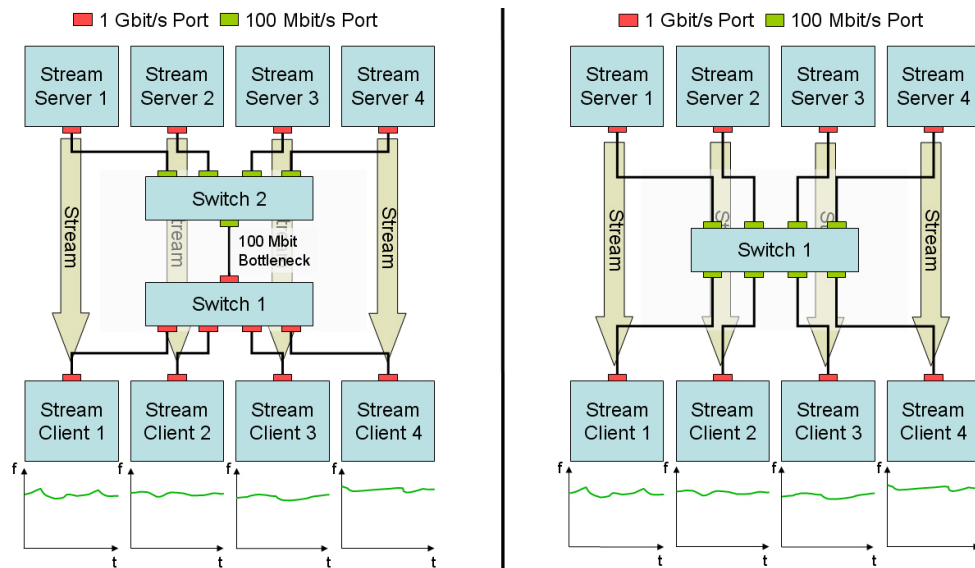


Figure 5.19: Network topologies for framerate measurements. Left image: with bottleneck; right image: without bottleneck.

to 32 *Mbit/s* per stream. The experiment showed that the RTP implementation which is used in the streaming service limits the framerate to 64 *Hz* and the framesize to 64 *kBytes* including the RTP header. Within these limits no stream can be established with a bandwidth higher than 32 *Mbit/s*.

Figure 5.20 shows for all four clients and all 81 settings the quotient of the measured framerate and the nominal framerate. Over the fully switched network, the average framerate measured in the streaming clients was in the worst case by less than 0.5% below the nominal value, which will for most applications be sufficient. The average framerate is a measure for the servers' capability to generate frames and the availability of sufficient network bandwidth. With 32 *Mbit/s* streams in a fully switched network the maximum network load was at 32%.

In order to investigate the behavior of the streams under a condition where the available bandwidth of the network is insufficient for transmission of all streams, a second series of experiments was conducted. Therein, a second network switch was added to the setup which introduced a 100 *Mbit/s* bottleneck through which all four streams had to pass (see Figure 5.19, left). The same 81 settings were run with this setup. The scatterplots in Figure 5.21 show for all four clients and all 81 settings the quotient of the measured framerate and the nominal framerate.

By definition, the quality of a streaming connection is not only characterized by the throughput, i.e. the average frame rate. In real-time conditions, clients rely on frames being delivered at regular intervals. The reliability of a streaming interface is characterized by the ratio of frames which are not delivered in time. The hardness of real-time conditions, i.e. the tolerance which is acceptable in the intervals between two frames, depends on the application. The acquired frame intervals were filtered according to different tolerances ranging from 1% to 50% of the nominal frame periods: frames which were not delivered within the tolerated delay were considered *dropped*. The *frame drop ratio* was calculated as a function of nominal frame rate and frame rate tolerance as the quotient of dropped frames and sent frames. The frame drop ratios for TiCoLi streams at framerates between 1 and 91 *Hz* are presented in Figure 5.22

Up to a framerate of 12 *Hz*, the frame drop ratio is below 0.1% even for a minimal tolerance. Streams with framerates of 24 *Hz* and higher showed a significant frame drop ratio at tolerance levels below 10%. Expressed in milliseconds, this means that all frame drop ratios were on an acceptable level when the tolerated frame delay was above 3 *ms*.

This observation can be explained with the sampling rate at which the client measured the frame intervals. The timing interval of Microsoft Windows XP which cannot be set to a period below 1 *ms* introduces a *deterministic jitter* to the handling of frames in the streaming interface as well as to the measuring process performed in the experiment.

At 24 *Hz*, e.g., the nominal frame interval is 41.7 *ms*. A tolerance level of 3% relates to a tolerated delay of 1.25 *ms*, which rounds to 1 *ms* in integer arithmetic. The sampling rate of the time measurement and the thread scheduler, which are both at 1 *ms* are too long for accurately generating a frame period of 1 *ms*. Based on the Nyquist-Shannon sampling theorem, the shortest integer period which can accurately

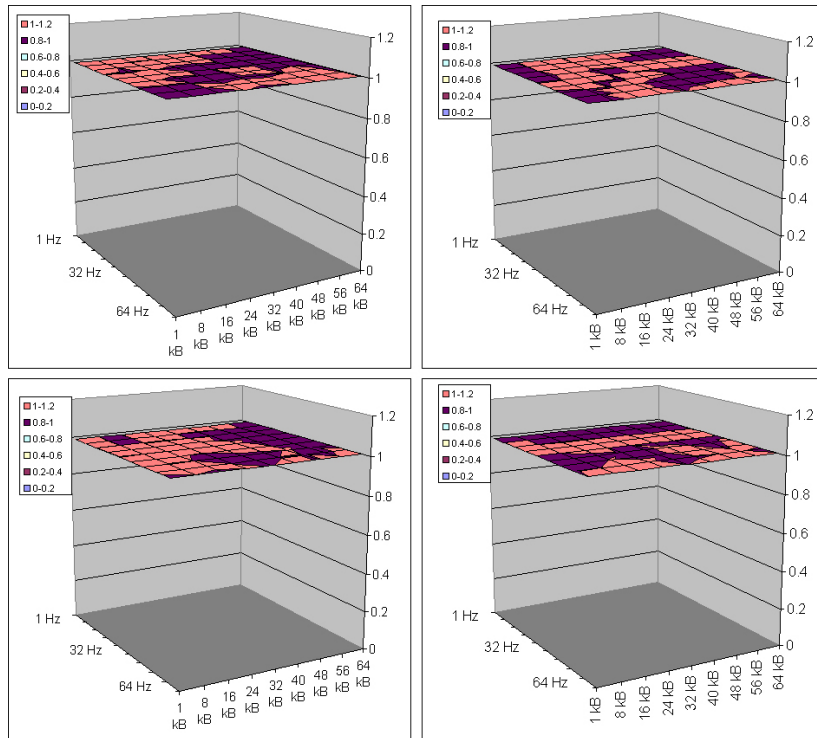


Figure 5.20: Average framerate divided by nominal framerate in fully switched network.

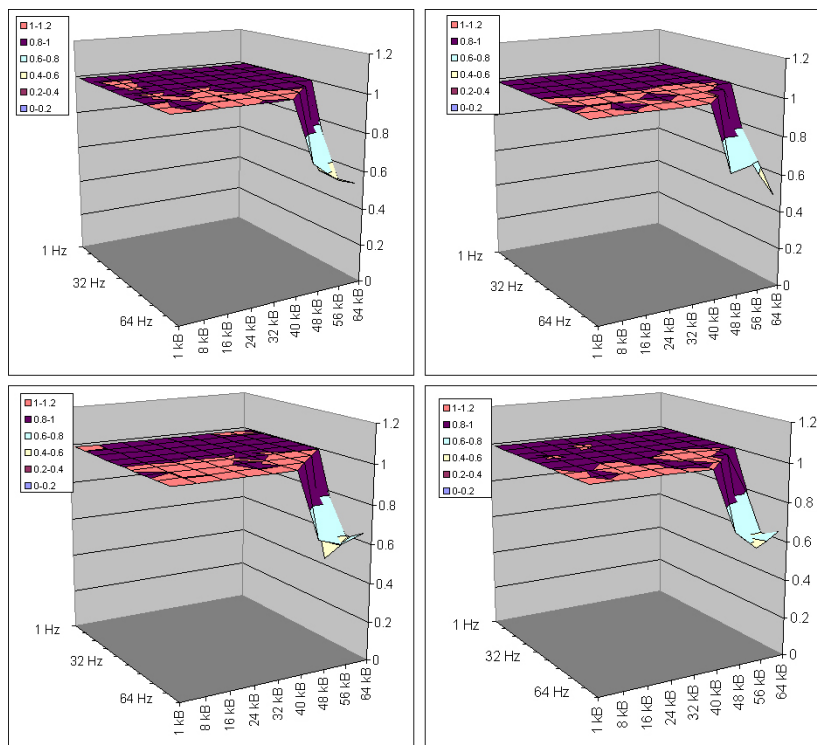


Figure 5.21: Average framerate divided by nominal framerate in a setup with a bottleneck.

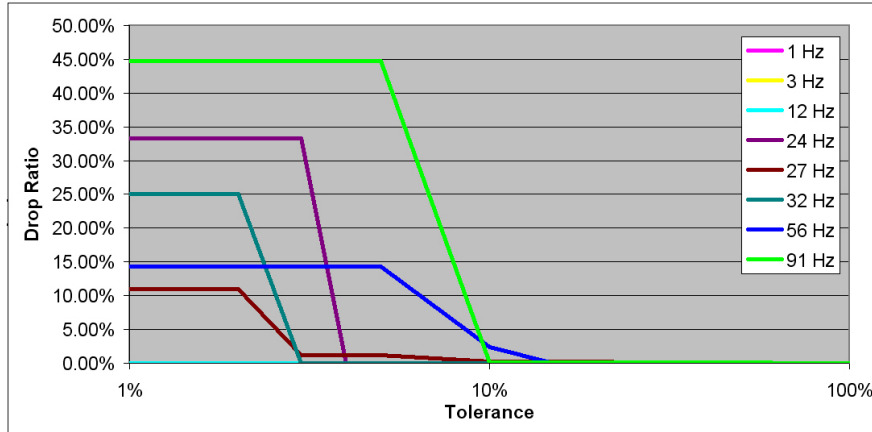


Figure 5.22: Frame drop ratio of TiCoLi streams with different framerates as a function of frame interval tolerance.

be generated is 3 *ms*. All frame drop ratios are well below 1% when applying a tolerance of 3 *ms*.

Discussion

The TiCoLi streaming service was tested with four streams which were transmitted through two different network setups. In a fully switched setup as well as in a case where the streams were sent through a bottleneck, the rate at which frames were received were acceptable as long as the bandwidth through each network segment was below the capacity of the segment. For the setup with bottleneck, the nominal frame-rates could not be sustained when four sending streams with a bandwidth of 25 *Mbit/s* per stream or more. With these measurements it is shown that the limit to the overall throughput of the TiCoLi streaming service is the bandwidth limit of the underlying network infrastructure. Within these limits, 98% of frames are delivered *in time* when allowing for a 5% variance in the frame intervals.

Another limitation of the streaming service is the maximum frame size of 64 *MByte* (including the RTP overhead) which is a limit set by the jRTP library. Should use cases require larger frames, improvements on the encoders and decoders of the TiCoLi could overcome this limit by decomposition of large frames into multiple packets which are reassembled after reception.

The presented performance tests were conducted with raw data streams as they are used, e.g., in tracking. The TiCoLi contains dedicated encoders and decoders for video streaming which apply on-the-fly mpeg4 compression. The computational complexity of these decoders and encoders potentially limits the maximum framerate at which the TiCoLi API can push packets into the RPT service on the server side as well as the rate at which the TiCoLi API is able to decode received packets to video frames. As soon as the video streaming module of the TiCoLi API on which Arun Voruganti is working is finished, the maximum throughput of this module requires further experiments.

5.4.2 Messaging Speed

In a second series of experiments, the throughput and delay of the messaging service was investigated. Therefore, two TiCoLi applications were implemented, the *MessageTimer* and the *MessageResponder*. The *MessageResponder* accepts messaging connects from any peer. The application contains a callback method for incoming messages which sends every incoming message back to the sender. The only alteration the *MessageResponder* does to the messages is the assignment of an actual time stamp.

The *MessageTimer* connects to the *MessageResponder*. Triggered by user input, it will start sending a series of messages to the *MessageResponder*. It sends one message and waits for the reply. The time which passes after calling the `MessageTo()` method on the TiCoLi API until the message callback function is called by the TiCoLi API delivering the response is measured on the scale of 1 *ms*. The measured periods represent twice the delivery time of the message through the network. This value is written into a log file before the next message is sent. After 100 log entries, the program terminates. The average and mean of the message delivery time were calculated from the log file.

The *MessageTimer* was implemented in two versions. The *StatusMessageTimer* sends small status messages with only a few hundred *kBytes* of data per message. With a network speed of at least 100 *Mbit/s*, the delivery of such small messages should actually happen instantaneously and the delivery period is therefore difficult to measure. Instead of measuring the delivery and return period for one message, the *StatusMessageTimer* measures the time it took to send, receive, send back, and receive 100 status messages.

The *PolydataMessageTimer* sends `PolyDataMessages` containing 100,000 points and 99,998 triangles. With three 64 *bit* floating point numbers per point and three 32 *bit* integer indices per triangle, each polydata message adds up to 3,600,042 *Bytes* including the header. Additional overhead has to be factored in from the TCP/IP over Ethernet network. The maximum segment size for data exchange through TCP/IP over Ethernet is 1,500 *Bytes*, including at least 40 *Bytes* of TCP header. The polydata message is split into at least 2,466 segments. Each segments adds at least 78 *bits* of overhead (≥ 40 *Bytes* TCP/IP headers, 26 *Bytes* Ethernet header, 12 *Bytes* Ethernet frame gap), resulting in an overall transmission of $\geq 3,792,390$ *Bytes* per message.

The efficiency of the message transfer was tested with both *MessageTimers* under various conditions. Similar to the streaming experiments, two setups were used (see Figure 5.23). In both setups, three streaming servers and three streaming clients were running streams at adjustable bandwidths in order to measure the effect of network load from other sources onto the transfer durations. The first setup consisted of a fully switched network, in the second setup a 100 *Mbit/s* bottleneck was introduced.

During all experiments, the network load of all channels was monitored on an additional computer which accessed the statistics of the network switch via SNMP. This served the purpose of validating the generated network load and as a means to iden-

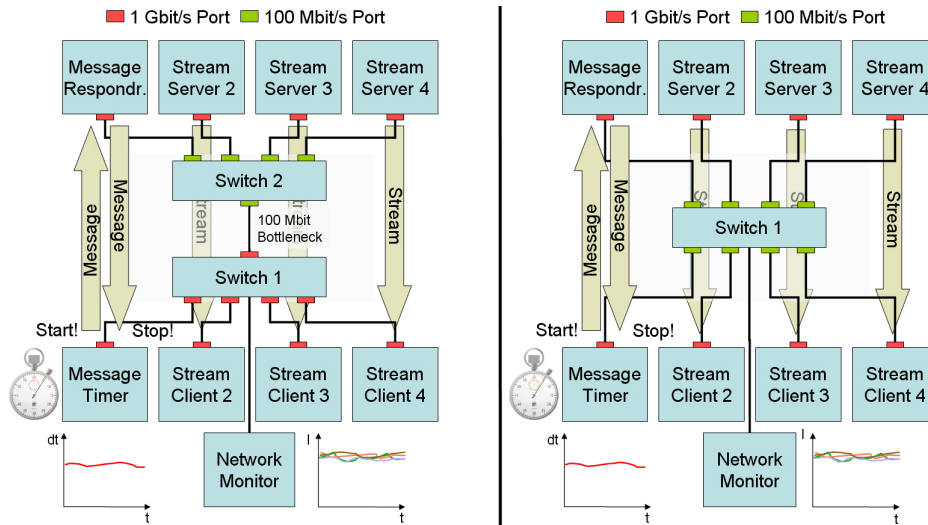


Figure 5.23: Network topologies for message speed tests. Left image: with bottleneck; right image: without bottleneck.

tify any unexpected network load which could influence the measured transfer durations. Different streaming parameters were adjusted to generate network noise between 24 kbit/s and 96 Mbit/s . In Figure 5.24, the average time for delivery of the polydata message (the mean of the measured delays for one stream configuration divided by 2) through the bottleneck with different network loads is shown.

The baseline duration for transmission of the polydata message through an unloaded network was 348 ms . Above a network load of 24 Mbit/s , The duration increased

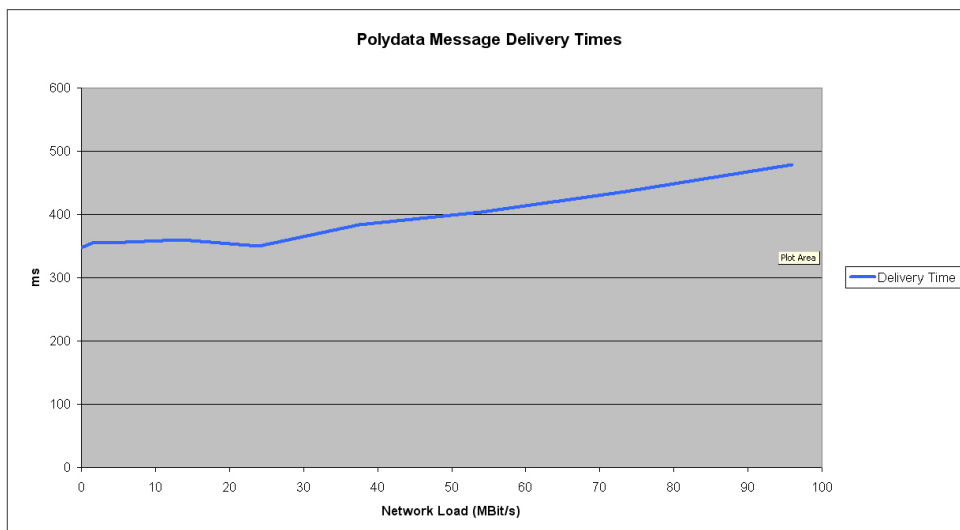


Figure 5.24: Measured message transfer duration for polydata messages as a function of network load.

roughly linearly with increasing network load when sending the streams and the messages through on bottleneck up to a value of 480 *ms* when competing with 96*Mbit/s* of streamed data on the network. In the fully switched case, the network load from the streams had no measurable impact onto the message transfer durations. For status messages, the average transfer duration was between 4 and 5 μ s per message - averaged over one hundred messages. This value was almost unaffected by the network load even up to 56*Mbit/s* and stays below 10 μ s even for very high network loads close to 100%.

Discussion

The network bandwidth was limited by the bottleneck to theoretical upper boundary of 100 *Mbit/s*. This resulted in a theoretical lower boundary for sending the polydata message of 289 *ms* per direction. For the status messages, the theoretical lower boundary for delivery through the network lay in the area of 1 – 10 μ s.

The sampling rate of the run time measurement was 1 *ms*. Regardless of the network load, the status messages were sent to the server and back in less than 10 μ s, which is regarded to be a good approximation for "instantaneous" message delivery.

With no network load interfering with the transmission through a 100 *Mbit* network, the polydata messages were delivered in approximately 348 *ms*, which exceeds the theoretical lower boundary by 59 *ms* or 20 %. This is owed to the fact, that either the Windows API, or one of the network hardware components limited the maximum bandwidth for one transfer to ≤ 90 *Mbit/s*. This was observed for all TiCoLi transmissions, but also for network folder access or other kinds of network transmissions. Under this premise, the lower temporal boundary for the polydata message to be transferred is 322 *ms*. The measured transfer delay exceeds this value by 26 *ms* or 8%. Considering that this period contains the transfer of the data to and from the networks card, as well as compiling and parsing of a 3.4 *MB* message, this delay is regarded to be acceptable.

5.5 Summary and Discussion

With the TiCoLi, a software library was presented which can be used by applications to establish sessions for exchange of data and commands through an IP - based network. The library implements the discovery of compatible devices in the network and the exchange of service descriptions between peers and the establishment of peer-to-peer sessions. With the TiCoLi, messages as well as streams can be exchanged between applications once a session is established. Applications can use the TiCoLi to grant peer applications access to internal methods and parameters.

A series of experiments was conducted with the aim to measure the capabilities and limitations of the library regarding the speed of data transfer and the level to which the streaming service complies to real-time conditions. The identified limits in stream and message data throughput and accuracy of frame timing were close to the bandwidth

limits of the underlying Ethernet network which was 100 *Mbit/s* and the timing resolution of the operating system which was 1 *ms*. The experiments revealed possible instabilities of frame rates when sending high-bandwidth streams and large messages through the same network segment.

Outlook

The TiCoLi is intended to be an evolving library which is continuously extended and adapted to the requirements of the applications which use it. As of autumn 2009, a state in the development is reached where the session initialization and the handshakes for service invocation are ready to use. The managers and the core of the TiCoLi which are responsible for the handling of all services are finished and went through an initial round of tests. The TiCoLi is already used in one project where an intraoperative modular setup is integrated with the TiCoLi (see Section 6.2). This project was intended as a test run of all services and marks an important milestone in the development of the TiCoLi.

With the core components ready, the TiCoLi can now only evolve based on the input of users. It is planned to release the library into the public domain as an open source project and to advertise its utilization in research projects. The following short-term extensions are currently planned for the TiCoLi:

Bandwidth management: Reduction of the impact of message transfer to the frame drop ratio of streams is the most pressing target regarding the performance of the library. Two approaches are being discussed. One possible solution would be to make use of the Quality of Service (QoS) bits which can be set in the header of an IP packet in order to prioritize the packet of other packets when handled by a network switch. In addition, the Resource Reservation Protocol (RSVP) could be included into the handshake performed to initialize a stream. This protocol allows two peers in a network to reserve a certain amount of bandwidth along one fixed route through the network. Both QoS and RSVP are not part of the original publication of the IPv4 standard. As it is not uncommon when dealing with internet standards, there exist differing implementations of both mechanisms in different networking hardware components and some routers or switches don't implement them at all.

The second approach is the integration of a central management node into the TiCoLi network which regulates the assignment of bandwidth on the application layer. The TIMMS Component Controller (TCC) which is developed by Stefan Bohn at ICCAS on the basis of the TiCoLi could play that role. A functionality is planned for implementation in the TCC which allows TiCoLi devices to request and reserve bandwidth for peer-to-peer communication. A mechanism in the messaging service is then required which limits message transfer to the assigned bandwidth.

Definition of additional frame and message classes: In its current version, the TiCoLi contains message types for the exchange of commands, images, triangular meshes, and status messages. The streaming service offers frame classes for tracking and video streams as well as an unspecific frame class which can be used to stream bulk data. In Chapter 3, requirements to the exchange of information between devices in

the OR were derived from surgical workflows. In order to comply with these requirements, at least a streaming service for biosignals is required. Additional use cases might require additional frame and message types, e.g. for streaming of compressed or uncompressed audio data in telemedicine.

Security features: an authentication mechanism during session initialization and a mechanism that validates the origin of a message or frame through a checksum or public-key cryptographic mechanism is considered. With these mechanisms, an application could restrict its services to a user group, or a group of devices. The uses cases and requirements for such mechanisms are not fully understood at present and additional research needs to be invested in this regard. One of the security aspects which need to be considered during session initialization is the authentication of the correct TiCoLi software versions in both peers in order to assure compatibility.

Scripted Device Descriptions: In the actual version of the TiCoLi, the service description for each service is generated and parameterized by the application. The developer of an application is required to hand-code these definitions in the source code of the application resulting in lengthy and obfuscated initialization routines. In order to facilitate the development process and the code quality of TiCoLi applications, a scripting mechanism is planned which outsources the definition of the service description to an xml-file which is parsed by the TiCoLi on startup. A graphical tool will assist the developer of a system in generating this file.

Chapter 6

Clinical Applications

In this chapter, the application of the proposed DICOM SOP classes and the TiCoLi infrastructure to integrate the modules for two CAS applications is presented. In Section 6.1, a planning system for transcatheter aortic root implantation is presented. The system is using DICOM services to retrieve image data from a PACS server and from an angiography systems. It utilizes the implant template query and storage services to obtain 3D models of stented valves from a repository. Planning results are exported using the surface segmentation storage and implantation planning SR document storage services. In Section 6.2, a CAS system for localization of a critical neuroanatomical landmark, the central sulcus, is presented. The application includes several pre- and intraoperatively used modeling systems which are interconnected using DICOM services as well as the TiCoLi.

6.1 Transapical Aortic Valve Implantation

Stenosis of the Aortic Valve (AV) is the most common acquired valvular heart defect with more than 12, 000 cases per year (2007) in Germany. Patients are usually between 60 and 80 years old [Morgan, Jr. *et al.*, 2001]. If untreated, aortic valve stenosis is attributed with a high mortality.

Surgical valve replacement is the only definitive therapeutic strategy in aortic stenosis, indicated in presence of severe symptomatic disease with a valve orifice area of 1 cm^2 or less [Bonow *et al.*, 2006]. Hospital mortality of AV surgery is reported in the literature between 3% and 4% in patients without cardiovascular co-morbidities [Bonow *et al.*, 2006]. Current conventional surgical techniques consist of partial or complete sternotomy (opening of the chest) with extracorporeal circulation and cardioplegic arrest. In parallel with an overall increasing life expectancy more and more elderly patients are being diagnosed with aortic stenosis. These patients have high operative risk levels. Besides age, there are additional perioperative risk factors as low ejection fraction, pulmonary hypertension, respiratory dysfunction or renal failure. These co-morbidities are associated with an increased perioperative risk, particularly for mortality.

Transapical Aortic Valve Replacement (TA-AVI) is an important treatment option for such high-risk patients. It is a minimally invasive technique for the deployment of stented bioprotheses, i.e. a prosthesis consisting of three leaflets crafted from bovine pericardial tissue that are sewn into a metal stent (see Figure 6.1). For implantation, the stent is crimped onto a catheter to a low profile. Through small incisions in the chest and the cardiac apex, the catheter is inserted into the left ventricle. The catheter tip is positioned within the aortic root, where the stent is expanded to deploy the valve. The procedure is performed on the beating heart, avoiding the use of extracorporeal circulation.

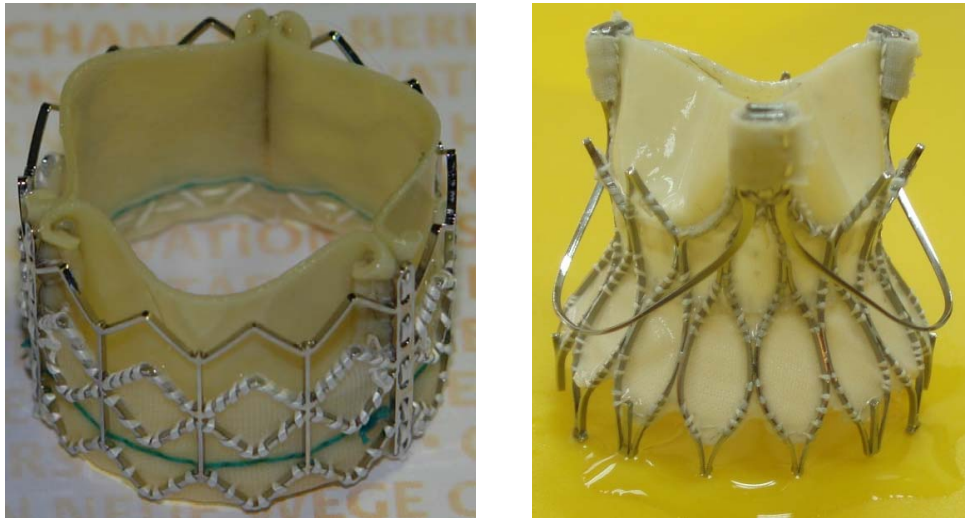


Figure 6.1: Sapien (left) and Embracer (right) valve implants.

In Figure 6.2, the top level workflow of TA-AVI is presented. It contains the following work steps:

- **Preparation:** During preparation of TA-AVI, the patient is anesthetized and a pigtail catheter for contrast agent injection is brought into the ascending aorta through a femoral arterial access. Through the femoral vein, a guide wire is brought close to the right atrium. This guide wire is important for quick start-up of the heart-lung machine if unforeseen events during the intervention require the conversion to a classic surgical approach with extracorporeal circulation.
- **Transapical Access:** Through a small incision on the chest, the intercostal space between the fifth and the sixth rib is exposed and extended to grant access to the apex. An incision on the apex is made and a so called purse-string suture [Kouchoukos *et al.*, 2003] is prepared for closure of the incision after the intervention. A sheath through which guide wires and catheters will be brought into the ventricle is inserted through the incision.
- **Apical Wire Placement:** A guide wire is inserted through the sheath into the

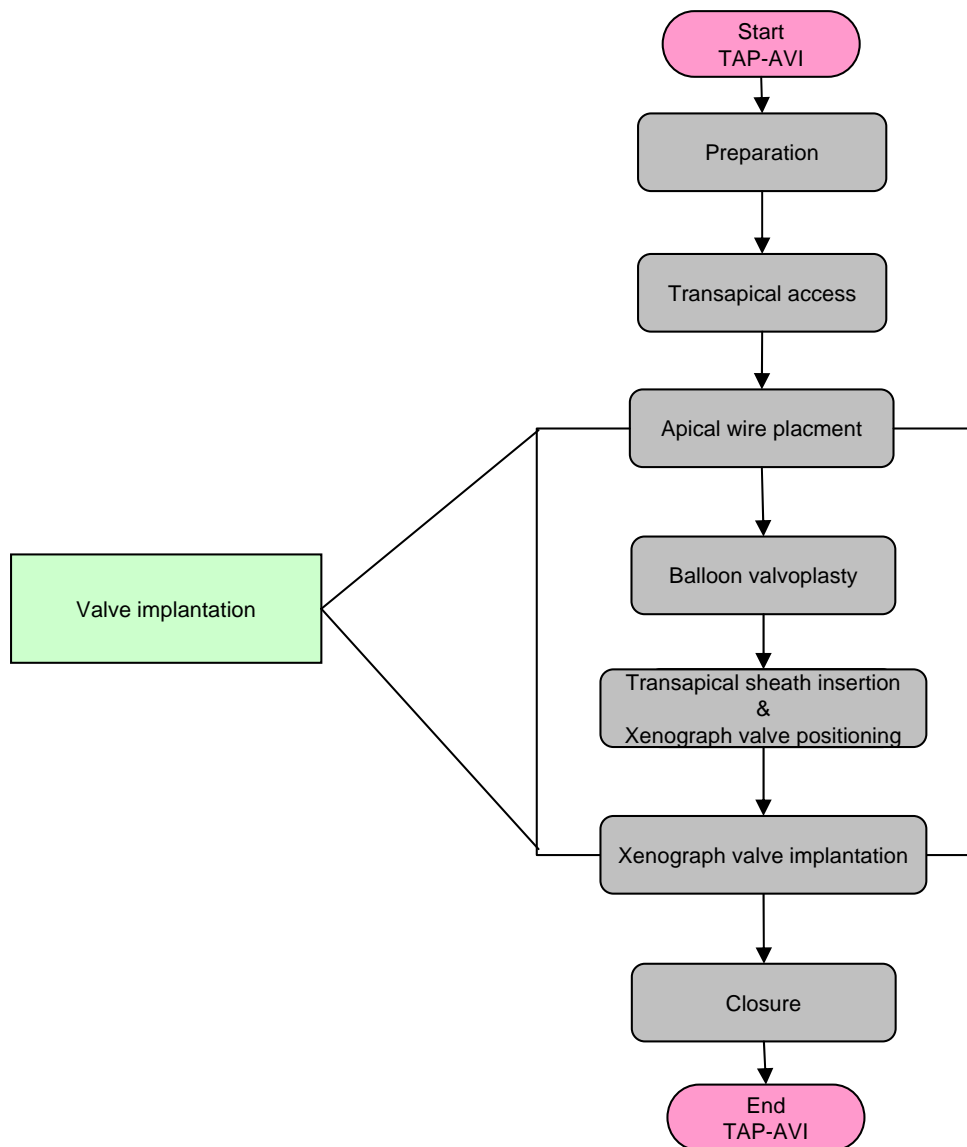


Figure 6.2: Top level workflow of TA-AVI.

aorta, usually up to well beyond the aortic arch. Along the guide wire, a balloon catheter is inserted and positioned inside the aortic valve.

- **Balloon Valvuloplasty:** Under rapid pacing (see below), the balloon on the tip of the catheter is inflated, dilating the aortic root and aortic valve.
- **Transapical Sheath Insertion:** The sheath through which the aortic valve implant will be delivered is inserted through the apex. Through the sheath, a catheter with the valve mounted on the tip is brought into the aortic root.
- **Xenograft Valve Positioning:** Under X-ray fluoroscopy guidance, the crimped

implant is positioned. When in place, the valve is deployed, either with another balloon dilatation or by releasing a self-expansion mechanism of the implant.

- **Valve Assessment:** With the means of contrast enhanced X-ray angiography and 3D+t Doppler ultrasound images, the functionality of the implanted valve is verified.
- **Closure:** When the valve is in place and the position and function of the implant have been checked, all sheaths and catheters are removed and the incisions are closed.

Different types of valves are available with different designs and different deployment mechanisms. For this work, two types of valves are considered which are described in the following paragraphs.

Sapien Valve

The Sapien valve consists of three xenograft leaflets which are sewn into a metal stent (see Figure 6.1, left). The stent and the valve are crimped to a low profile around a balloon catheter. This catheter is steered into the aortic root. The valve is placed in the aortic annulus. The diameter of the stent is selected to be $1\text{ mm} - 2\text{ mm}$ larger than the diameter of the ventriculoarterial junction. This leads to a tight press fit between stent and aortic root. The metal of the cage is engraved into the vascular and ventricular wall. Once the stent is inflated, it cannot be retraced or repositioned. (see Figure 6.3). For placement of the valve, there exists a very short target area between the proximal end of the aortic root and the level of the coronary ostia

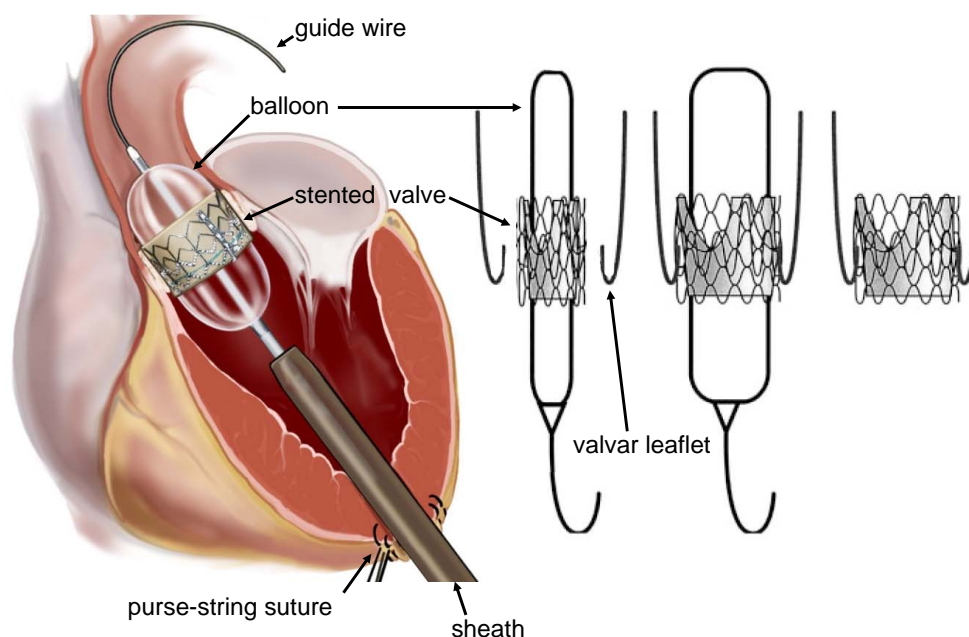


Figure 6.3: Transapical implantation of the Sapien valve.

Embracer Valve

The Embracer valve consists of a soft metal stent carrying the actual xenograft valve leaflets. Attached to the stent, three Nitinol [Buehler *et al.*, 1963] support arms reach down from the commissures. Like the original valvar leaflets, the xenograft leaflets are attached to the stent in a semilunar fashion. Nitinol is a highly elastic alloy from Nickel and Titanium. When the valve is crimped to low profile for insertion through the apex, the support arms are deformed. They act like springs, when the sheath holding the stent together is removed and unfold the valve to its original shape. After implantation, the support arms are embedded inside the aortic leaflets (see Figure 6.4).

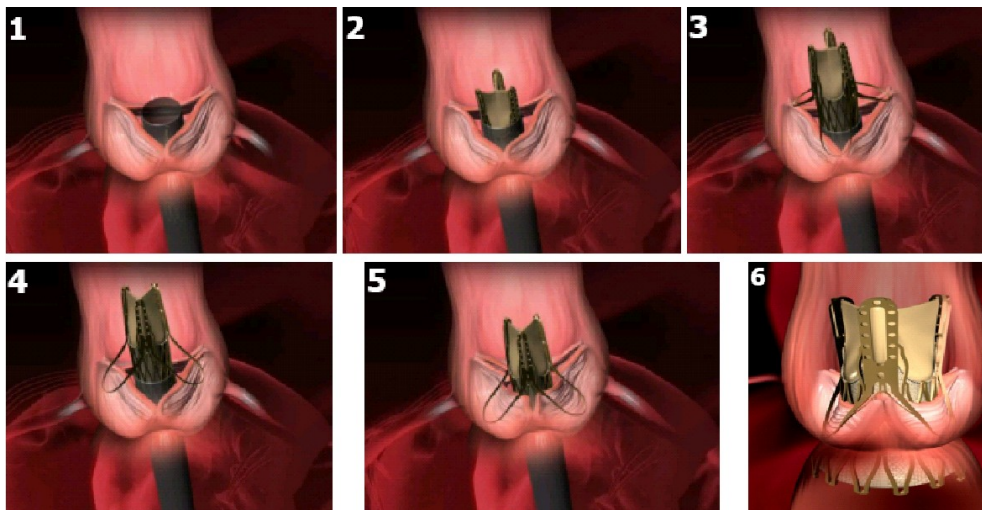


Figure 6.4: Transapical implantation of the Embracer valve (images courtesy Ventor Medical Technologies).

Limitations

TA-AVI has been performed on high-risk patients more than 1,000 times in selected medical centers with a success rate well above 90%. Nevertheless, there are some difficulties about this technique which makes it hard to learn and perform. Literature names a number of adverse events and complications which can occur during TA-AVI, including valve dislocation, coronary occlusion, paravalvular leakage, stress-induced disorder of stimulus conduction, and vascular or septal injuries [Walther *et al.*, 2008; Al Ali *et al.*, 2008; Clavel *et al.*, 2009; Wong *et al.*, 2009; Himbert *et al.*, 2009; Al-Attar *et al.*, 2009].

Accurate placement of the implant is an important factor to reduce the incidence of these events. Therefore, positioning of the stented valve before deployment of the valve is the most critical work step of the intervention. The following limitations were identified regarding this work step:

- The angiography sequences describe the geometric situation insufficiently due to the fact that they consist of projective 2D images.

- The X-ray contrast of the myocardium, the vessels, the valve and the blood is very low. In particular, the coronary ostia are hard to perceive. Injection of contrast agent solves this issue for a few seconds, but the contrast agent affects the patient's kidneys and should therefore only be applied at a minimal dosage.
- The exact shape of the valve after the inflation is not visible in the X-ray images. The folded metal stent is clearly visible even without any contrast agent, but it will change its shape during inflation. The surgeon has to estimate the resulting shape.

In addition to the intraoperative limitations regarding valve positioning under image guidance, the selection of the optimal implant is important. It is expected, that a number of valve types in different shapes and sizes will be available for TA-AVI in the near future. Similar to implantation planning routines in orthopedic and other fields of surgery, a preoperative planning procedure needs to be defined according to which the selection for an implant type and size can be made.

6.1.1 Computer Assisted Transapical Aortic Valve Implantation

Systems for enhanced intraoperative imaging, computer assisted planning, and computer assisted valve placement are being developed by several vendors and research centers. From Siemens Healthcare, a volumetric angiography system is available which enables intraoperative 3D imaging of the aortic root. A digital planning system was developed which extracts a surface model of the aortic root from the volumetric angiography images and virtually places a three-dimensional implant template inside the model (see Figure 6.5) [Gessat *et al.*, 2009]. It is intended to work together with an intraoperative image based tracking system which is being developed by Mohamed Karar [Karar *et al.*, 2009]. This system intraoperatively tracks the position of the aortic root and the stented valve and will be able to compare the actual position of the valve with the planned target position.

Commercial systems with similar functionalities are available from 3mensio Medical Imaging [3Mensio, 2009] and Paieon Inc. [Paieon, 2009]. Neither the 3mensio nor the Paieon system is able to import modeling or planning results in other than a proprietary format and do not produce output data in a standardized manner. Both systems can be regarded as monolithic. The ICCAS approach is based on open interfaces with the aim to design and implement independent systems for treatment planning and treatment support and to embed these systems into an open infrastructure.

6.1.2 Infrastructure

The planning software as well as the intraoperative guidance software is built for application in the OR. Both systems need to exchange data with intraoperative image

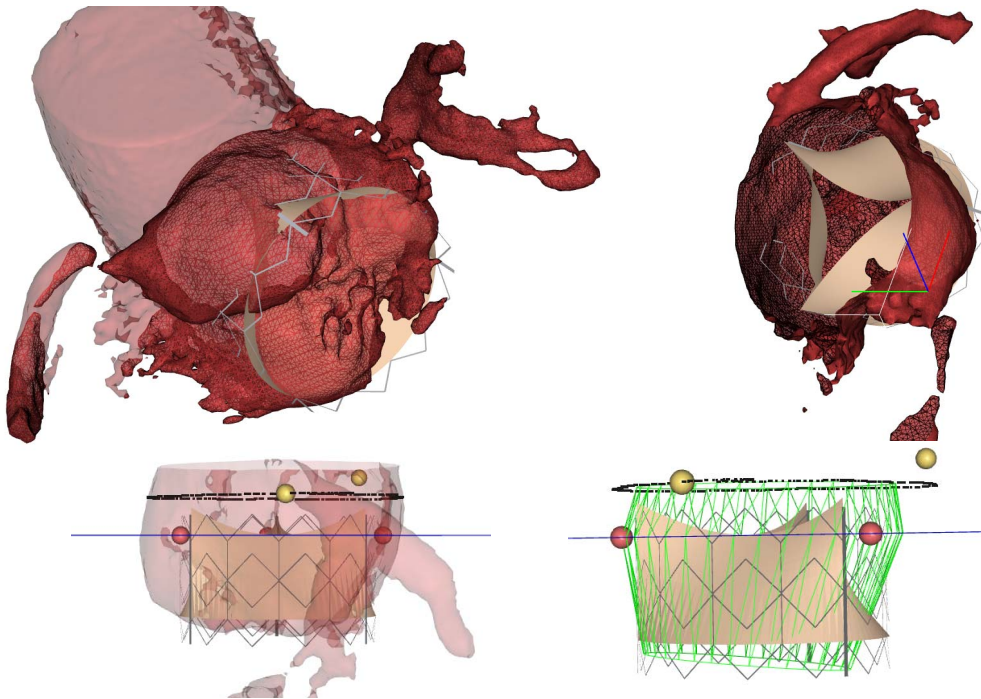


Figure 6.5: Implantation planning with Sapien implant template.

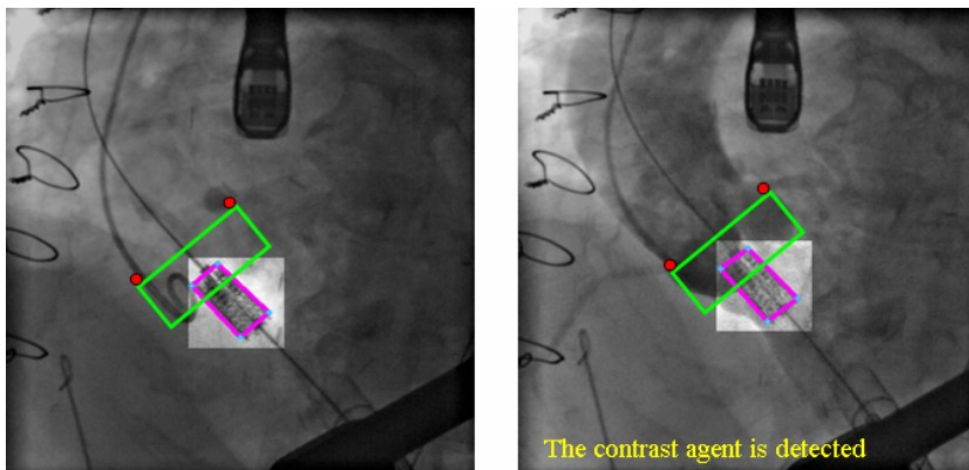


Figure 6.6: Intraoperative image-based tracking of the coronary ostia and the stent [Karar *et al.*, 2009].

sources and displays. The data flow diagram in Figure 6.7 summarizes the data exchange between the modules. Both DICOM and the TiCoLi can be applied to implement this dataflow.

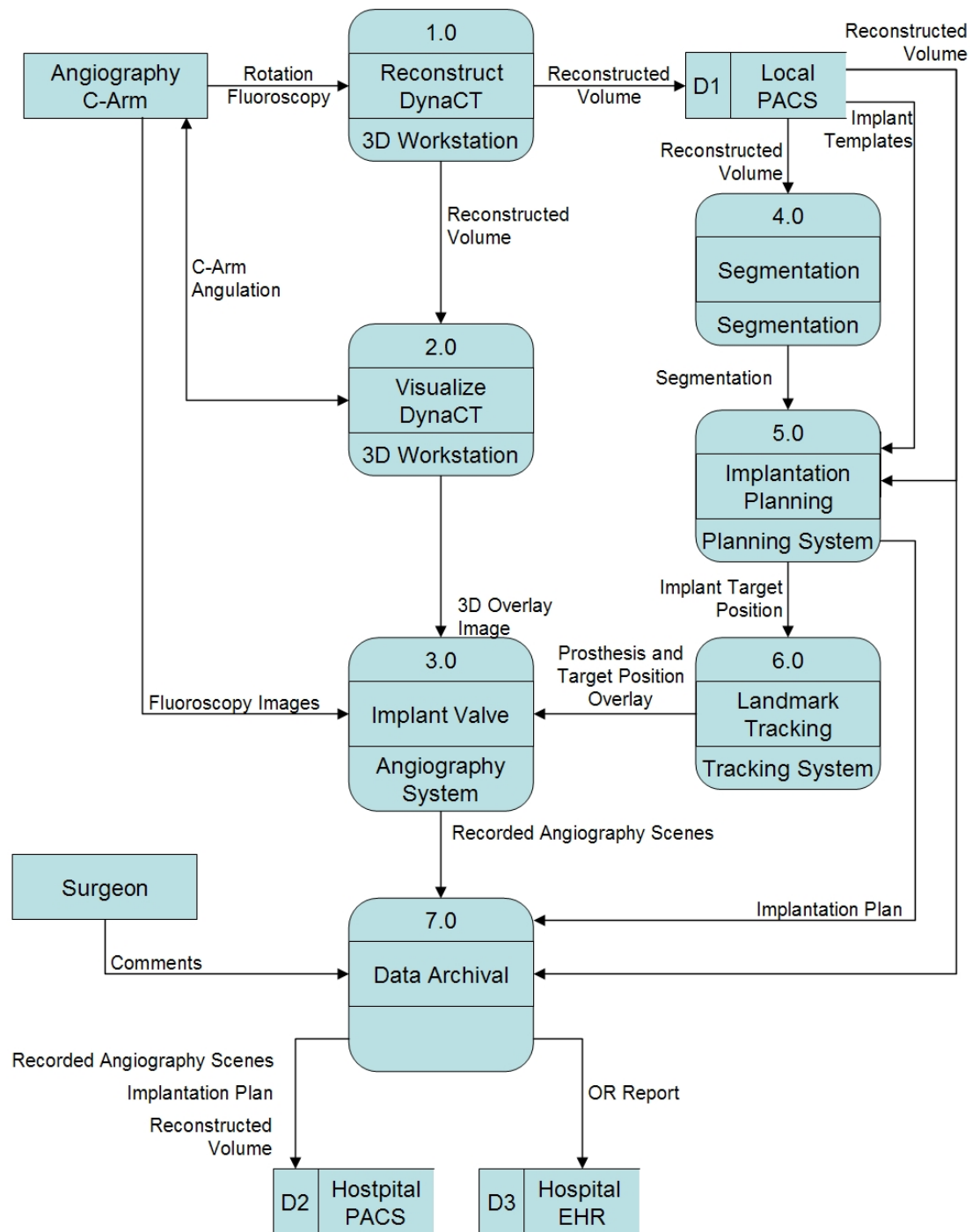


Figure 6.7: Proposed dataflow diagram for computer assisted TA-AVI.

Data exchange during planning phase

For exchange of large datasets, implant templates, and implantation planning results during implantation planning, DICOM transfer services are employed. This allows for long-term archival of the results on a PACS server after the intervention and made access to the radiology images from the DynaCT workstation possible without any adaptations on the workstation. Table 6.1 provides an overview of the SOP classes which are utilized in the setup for data exchange between the workstation performing the volume reconstruction, the planning and modeling application, an implant template repository, and a local PACS server.

Parameter	Type	Content / Description	Transfer Service
Angiography	Image	Recorded X-ray angiography scenes.	X-Ray Angiographic Image Storage
DynaCT	Image (3D)	Reconstructed volume showing the morphology of the aortic root	X-Ray 3D Angiographic Image Storage or CT Image Storage
Valve Model	Surface	Implant Templates used for planning	Generic Implant Template Storage
Segmentation	Image	Segmented shape of the aortic root	Segmentation Image Storage
Surface Segmentation	Surface	Segmented surface of the aortic root	Surface Segmentation Storage
Landmarks	Geometry	Points and planes representing relevant features of the aortic root	Spatial Fiducials Storage
Registration	Matrices	The transformations which register the implant template with the patient data	Spatial Registration Storage
Implantation Plan	Report	Collection of the implantation planning results	Implant Planning SR Document Storage

Table 6.1: DICOM SOP classes for data exchange in computer assisted TA-AVI.

The planning software implements both DICOM SOP classes which were presented in Chapter 4. The surface segmentation which is derived from the volumetric angiography image is exported to a PACS server with the surface segmentation storage service. The implant templates are encoded as implant template IODs. The software uses the DICOM toolkit DCMTK [DICOM@OFFICE, 2005] for communication with DICOM nodes and for parsing of DICOM instances.

In order to test the integration of the modules, an experimental PACS server was developed which implements the query and retrieve SOP classes for both the implant template IOD and the surface segmentation IOD. Several open source PACS servers

were available for this attempt. The inclusion of the surface segmentation IOD into the DCM4CHEE server [dcm4chee, 2005], for example, did not require any alterations on the server other than adding the SOP class UID to its registry. For implant templates, the integration would have required intensive alterations on the internal database structure on which the DCM4CHEE server is based. The problem was that the server would not accept any instances which do not contain a patient identification which is mandated by the database in every entry. The DICOM toolkit was used to implement a minimal implant template C-Find / C-Move / C-Store service class provider which acts a DICOM implant template repository.

Digital Aortic Valve Implant Templates

The implant template SOP class (see Section 4.3) is utilized to represent the shape and relevant properties of the aortic valve implants. In order to create DICOM instances compliant with the specification from that supplement, surface models of the implants were transformed into DICOM encoding and the additional information required to fully specify the templates was acquired by manual measurement and from the product catalogs.

Both available implant types were modeled. The planning landmarks which were added to the implant templates were selected according to the anatomical landmarks according to which the implants are intraoperatively positioned.

The implant templates which describe the Sapien valves contain the following planning landmarks:

- The longitudinal axis of the implant is encoded as a line landmark \mathbf{a}_f .
- The distal rim of the implant is marked by three point landmarks \mathbf{r}_{di} .
- Each commissure of the valve is marked by a point \mathbf{c}_{vi} .
- The proximal rim of the implant is marked by three point landmarks \mathbf{r}_{pi} .

The following planning landmarks are provided in the implant template instance of the Embracer implant

- The bottoms or sinuses of the valve's support arms \mathbf{b}_{vi} .
- The tips of the struts \mathbf{s}_i .
- The longitudinal axis pointing is encoded as a line landmark \mathbf{a}_f .
- The proximal rim of the implant is marked by three point landmarks \mathbf{r}_{pi} .
- The distal rim of the implant is marked by three point landmarks \mathbf{r}_{di} .

Figure 6.8 shows both templates and the planning landmarks.

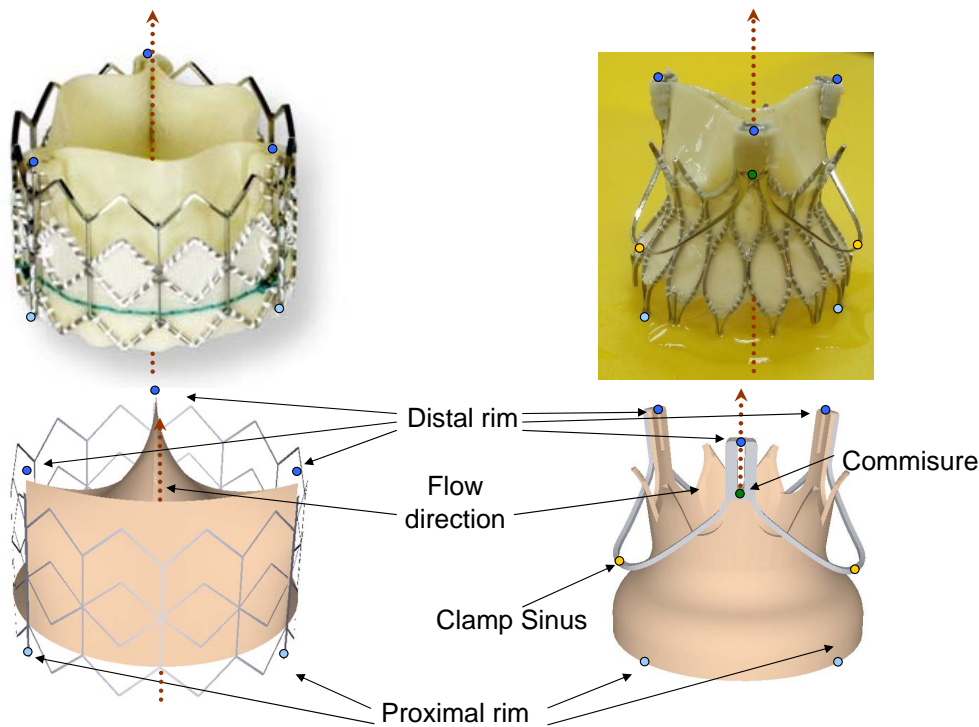


Figure 6.8: Digital aortic valve implant templates with planning landmarks (Photographs courtesy of Edwards Lifesciences and Ventor Medical Technologies).

Data exchange for image guided implantation

During the intervention, when the image-based tracking systems track the current location of the stent and the target regions, real-time exchange of video data and tracking coordinates is required. The streaming service in the TiCoLi could be used for this purpose. The video signal from the angiography C-arm therefore needs to be captured by computer through a frame-grabber which is connected to one of the video outputs of the device. The C-arm's acquisition parameters which are required for 2D-3D registration, such as the angle of projection and magnification factor, can also be sent to the TiCoLi network with the streaming service. Therefore, an available software interface for reading the device parameters from a dedicated bus would need to be encapsulated into a TiCoLi application. At present, the integration of these modules can only be presented as a concept, since the tracking module has not yet reached real-time capability and is therefore not ready for integration and clinical use.

Data exchange during reporting and storage of results

After the intervention, all the data which seems relevant for long-term follow-up and documentation in the patient's health record can be sent from the local PACS to the hospital's central PACS. This requires a network connection between both domains and the necessary access rights for the OR devices and personnel on the PACS server.

6.2 Cortical Stimulation and Mapping

Brain surgery, especially brain tumor resection, is a task which presents various challenges to a surgeon. One such task is the confident identification of critical areas of the cerebral cortex. Intentional or unintentional injury of the motor cortex, i.e. the region which is responsible for generation of impulses controlling the execution of voluntary muscle motion, would set the patient at risk of potentially irreversible mobility impairments. To avoid unintentional damage, intraoperative localization of critical areas is an important task during surgical resection of tumors, especially tumors which are close to the motor cortex. Besides indirect methods, such as functional MRI [Beliveau *et al.*, 1991] in combination with Diffusion Tensor Imaging (DTI) [Le Bihan *et al.*, 2001] which are employed preoperatively to identify the functional areas in the surroundings of the tumor, there exist direct methods for intraoperative localization of certain functional areas. One is surgery with local anesthesia, where the patient is sedated but awake and asked to perform tasks which are known to be affected by neurons in the cortex areas the surgeon wants to identify. For interventions close to regions which are associated with visual perception or word formation, the patient is asked to name objects which are presented to him on picture cards [Meyer *et al.*, 2001]. Such procedures expose the patient to high levels of stress but are sometimes the only viable approach to guide the surgeon past critical structures involved, e.g., in the neural process of speech.

The motor cortex can be localized through neurophysiological measurements. In open brain surgery, Somatosensory Evoked Potentials (SSEPs) are used to identify cortex regions. Thereby, electrical stimuli are induced at the peripheral nerves on the extremities. Electrodes which are placed on the cortex measure the electrical response of the cortex cells [Woolsey *et al.*, 1979; Gregorie & Goldring, 1984; Eisner, 2001; Romstock *et al.*, 2002].

6.2.1 The Central Sulcus

The central sulcus, i.e. the fissure on the cerebral cortex between the precentral and the postcentral gyrus (see Figure 6.9), is an important landmark in brain surgery. It separates two very important functional areas: the primary motor cortex which is responsible for voluntary motion and the primary somatosensory cortex which is responsible for haptic perception [Seeger & Zentner, 2002].

The anatomical localization of the central sulcus is well described [Gray, 1918; Trepel, 2004; Dützmann, 2009]. Nevertheless it can be difficult during brain surgery to identify the central sulcus among the sulci which are visible in the situs. One reason for this is that only a small portion of the cortex is exposed which makes it harder for the surgeon to relate the visible landmarks with the overall structure of the brain. The other reason is that due to tissue shift which is induced by a nearby tumor, the localization and course of the central sulcus in one particular patient can diverge strongly from the average expectation. In Figure 6.10, a typical view through the OR microscope

onto the cortex is shown the dashed line identifies the central sulcus among the sulci which are visible in the patient.

6.2.2 Localization of the Central Sulcus

For the intraoperative identification of the central sulcus, a special measuring method is employed after the skull and dura are opened. It makes use of an effect called *phase reversal* which can be observed across the central sulcus: When evoking an SSEP on an extremity, a response can be measured on the somatosensory and, due to electrical feedback effects, also on the motor cortex. The waveform of the evoked potentials is received *inverted* in the motor cortex (see Figure 6.11). Linear arrays of electrodes, called *grid electrodes* are placed directly on the cortex to measure the reaction potentials induced by the SSEPs. In Figure 6.10, the grid electrode is visible on the cortex. Between the electrodes which are on either side of the central sulcus, a phase reversal can be measured. The identification of the phase reversal is done by a neurologist. The transfer of the measuring results to the situs is done mentally by the surgeon.

In IGS systems, such as the Brainlab Vector Vision[®] or the Localite Neuronavigator systems, preoperative MR images are used to support the surgeon in localizing the central sulcus. The reliability of these systems is impaired by the deformation of the brain during surgery ("*brain shift*"). Three major reasons for brain shift are [Trantakis *et al.*, 2003]:

- The way the patient is laid, seated or otherwise positioned on the OR table is usually different from the position he or she was in when the images were acquired. This leads in a different resulting gravity vector for the brain.
- The stabilizing effect of the cerebrospinal fluid (CSF) which usually fills the ventricular system, i.e. a system of cavities inside the brain, decreases when the dura is opened and parts of the CSF flow out.
- The internal tension of the brain tissue which is compressed due to the growth of the tumor is released when the skull is opened and the brain expands.

Limitations

With the available technology, neurosurgeons and neurologists are able to identify areas on the exposed neurocortex. There is no established technique which combines the sensory output with the, highly relevant, information about the location of its acquisition. The task of correlating the information with the preoperative findings in an MR image, in order to, e.g., validate a plan or estimate brain shift, is left to the surgeon who has to mentally translate between both worlds – the real-world situs and the virtual planning model.

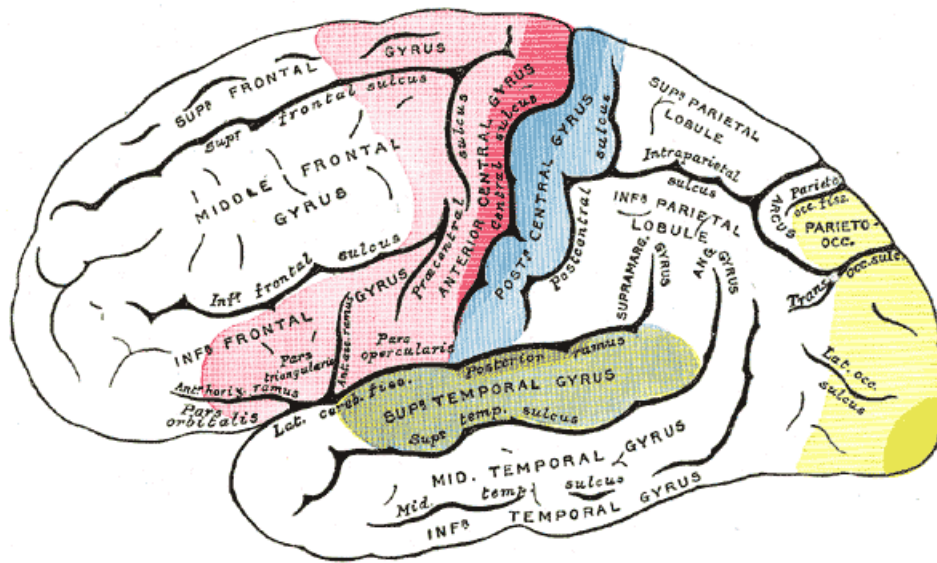


Figure 6.9: Principal regions of the cortex. The central sulcus separates the motor cortex (red) and the somatosensory cortex (blue) [Gray, 1918].

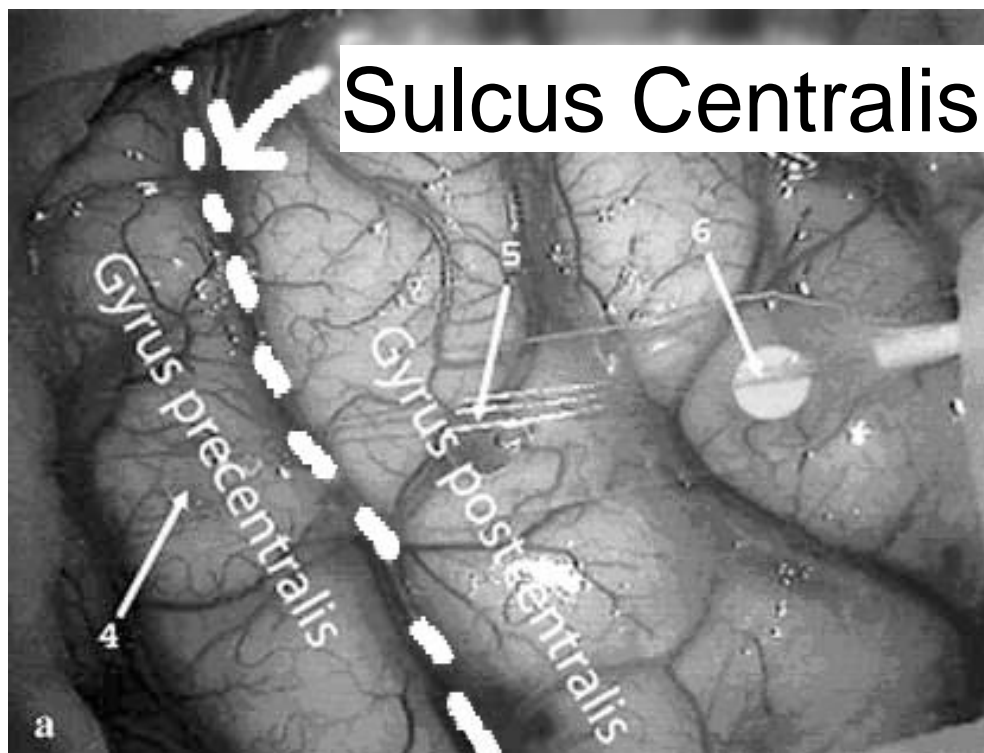


Figure 6.10: Intraoperative view onto the cortex.

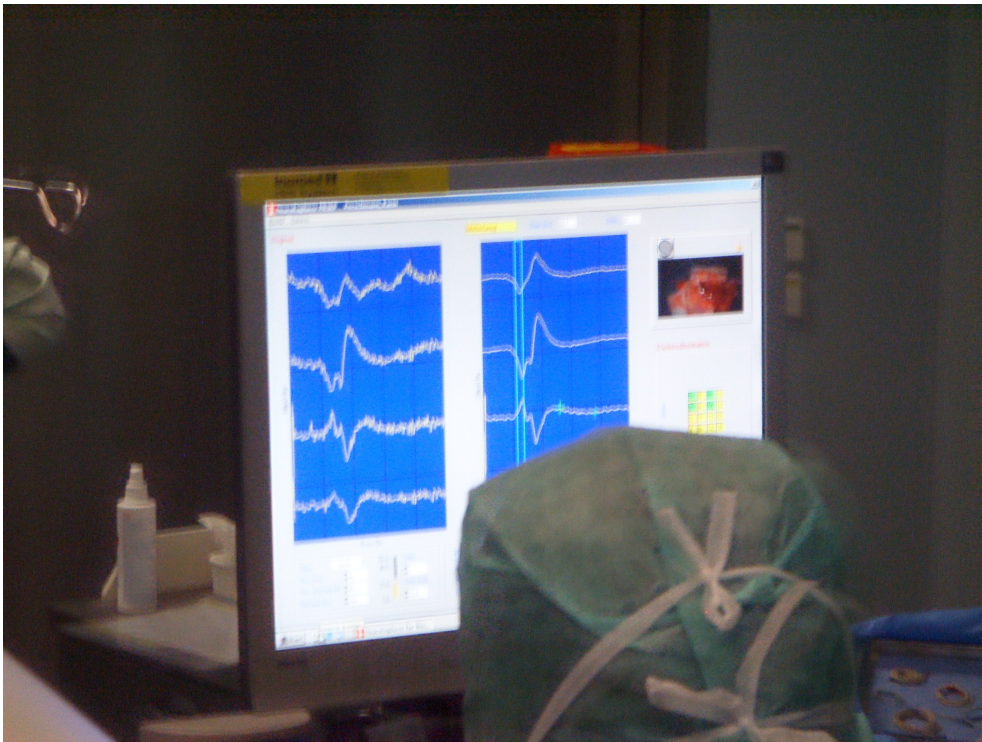


Figure 6.11: Acquired action potentials from a 4-electrode grid. Phase reversal is between the second and the third electrode.

6.2.3 Intraoperative Mapping of the Central Sulcus

With the existing surgical navigation technology, geometric information about cortex areas obtained from preoperative images can be spatially related to real-world coordinates in the OR, neglecting tissue shift. With the existing neurophysiological sensors, areas on the exposed cortex can be identified *in situ*. In this section, a system is described which combines both technologies in order to provide the neurosurgeon with an intraoperative generated model of the central sulcus which is visualized in combination with a preoperative surface model of the neurocortex. The system allows for accurate distinction between the pre- and postcentral gyrus and can be used to locally estimate the accuracy of navigation based on preoperative data. The system is developed at ICCAS in Leipzig including the works of Daniel Streitbürger, Dr. Rafael Mayoral, Stefan Franke, and the author as well as of Prof. Jürgen Meixensberger and Dr. Christos Trantakis from University Hospital Leipzig. The concept of the work and first results were presented in 2009 [Streitbürger *et al.*, 2009].

Functionalities and Modules In Figure 6.12 the setup of the system for intraoperative mapping of the central sulcus is presented. The core component of this setup is the *NeuroMapper* application. It collects tracking information about the position of the patient and a handheld pointer. The handheld pointer is utilized by the surgeon to

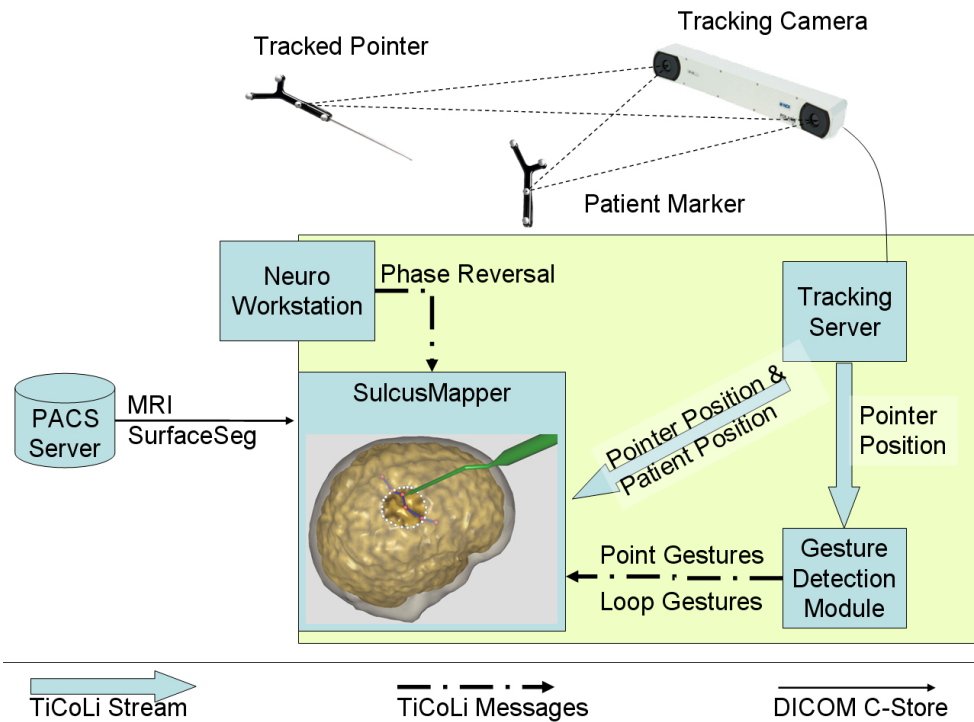


Figure 6.12: Intraoperative system setup for sulcus centralis mapping.

indicate the positioning of the grid electrode used to pick up the SSEP signals on the cortex. This information is combined with input from a diagnostic workstation about whether and between which electrodes phase reversal was observed by the neurologist or a trained nurse. In Figure 6.13, a schematic drawing visualizes the process of point wise identification of the central sulcus and piecewise linear interpolation of its course is shown. The tracking algorithm is able to extrapolate the point on the cortex which corresponds with a phase reversal even if the grid was pushed forward underneath the

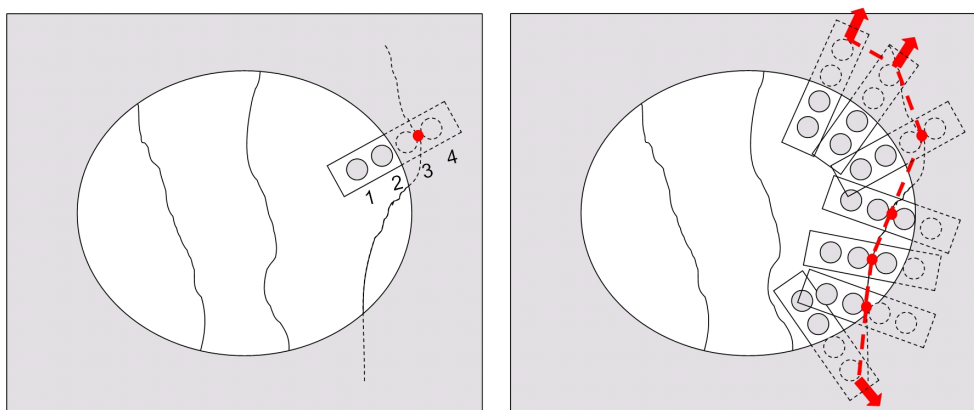


Figure 6.13: Localization of the grid electrode and the central sulcus.

skull beyond the rim of the trepanation. An example for the intraoperative tracking result acquired with the presented system is shown in Figure 6.14

The *SulcusMapper* has access to a PACS server from where it receives preoperative MR images and models of the patient's anatomy (see Section 6.2.4). Intraoperatively, the *SulcusMapper* interacts with three additional software modules which are running on separate PCs:

- The *Gesture Detection Module* is a generic low-level module which is used to monitor the motion of a tracked object, e.g. the handheld pointer, and to detect gestures which are performed with the device. The module is able to connect to tracking data servers in the network, monitor one or more of the objects a server tracks and notify a third application, in this case the *SulcusMapper*, when a gesture was performed. In its current version, the gesture detection module is able to detect two gestures which are sufficient for the presented setup.

The first gesture is used to indicate a certain point in space, e.g. a landmark on the patient or on the grid electrode. The gesture detection module assumes a point gesture every time a monitored object was kept still for a certain period. The length of this period can be configured.

The second gesture is a closed loop which can be utilized to circumscribe a region. A closed loop gesture is assumed by gesture detection module every time the path of a tracked object self-intersects. A lower boundary can be set for the minimal trip length of the circumscribed loop in order to filter out unintended

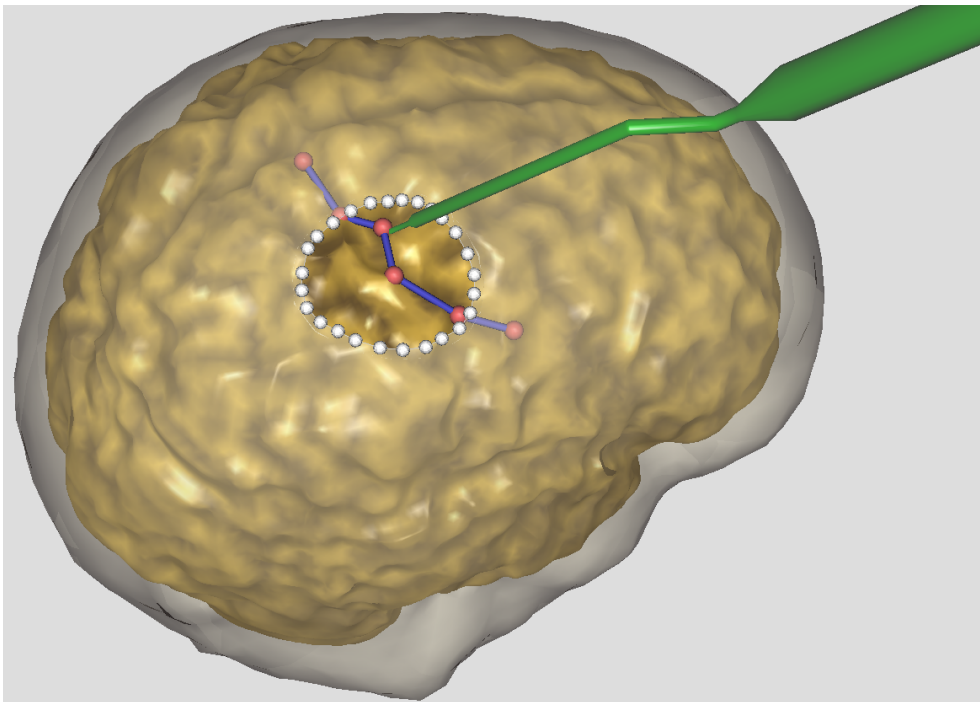


Figure 6.14: Visual representation of the central sulcus and the trepanation in the *SulcusMapper*.

small loops due to tremor during point gestures.

The gesture detection module is initialized and stopped by a peer using the remote method service of the TiCoLi. Two methods are made available which are called by a peer to start and to stop a monitoring job. Detected gestures are sent to the peer which initialized a job with a TiCoLi `PolydataMessage`.

The internal logic of the gesture detection module is presented in Appendix D.3.

- For online visualization of the pointer position in relation to the preoperative and intraoperative models, the unprocessed tracking stream is received from the same server which is the source of tracking data for the gesture detection module. The *Tracking Server* is a generic module which can access tracking cameras of different kinds and provide the tracking coordinates they deliver through a uniform interface. The tracking server offers TiCoLi streams for every object the attached navigation system is tracking. In the presented setup for mapping of the sulcus centralis, the tracking server offers two streams: one for the hand-held pointer and one for the patient tracker which is fixed at the patient's head. The *SulcusMapper* connects to both streams, whereas the gesture detection module only connects to the stream with the position of the handheld pointer.
- The neurophysiology workstation from where the result of the SSEP measurement is sent. In the present state of development, the workstation is not directly integrated into the setup. Instead, the neurophysiology nurse has to manually input the result of the measurement. The information is sent to the *SulcusMapper* via the messaging service of the TiCoLi.

The data exchange between all modules through one Ethernet network was possible without any limitations regarding bandwidth, data losses, or difficulties during auto-configuration. During all tests in the laboratory as well as in the operating room, both tracking streams were sent and received at the nominal framerate of 20 *Hz* which is the framerate at which the NDI Polaris tracking system generates coordinates. Network latencies were not perceivable. The streams were in no measurable way affected by the messages which were sent from the gesture detection module. During setup, no user interaction is required: the *NeuroMapper*, the gesture detection module, and the streaming server are plugged into the network which assigns IP addresses to all three devices. The gesture detection module automatically establishes a connection with the streaming server. The *NeuroMapper* automatically identifies the streaming server and the gesture detection module and registers itself as a client with both peers.

6.2.4 Preoperative Model Generation

In order to extrapolate the shape of the flexible electrode as it is pushed forward underneath the skull, and for visualization of the sulcus mapping result, the *SulcusMapper* requires a surface model of the neurocortex and the skull. These surfaces are preoperatively extracted from a T1 weighted MRI image with a segmentation tool published by the FMRIB Center of the University of Oxford, called the *Brain Extraction Tool* (BET)

[Jenkinson *et al.*, 2005]. The BET runs in a Linux environment and requires images in the NifTI-1 file format [NifTI, 2005] as input and generates a triangular surface which it stores in the vtk file format [Schroeder *et al.*, 2004].

In order to integrate the BET into a perioperative workflow with minimal user interaction, the *Brain Extraction Service Provider* was developed. The software is intended to run as a background process on a server which automatically starts the extraction of the cortex model from MR images as soon as they are available. The system is able to send and receive DICOM as well as TiCoLi Messages. The BET service provider interacts with a second module, the *BET Server* which runs on the same hardware as the *Brain Extraction Tool*. It exchanges TiCoLi messages with the *Brain Extraction Service Provider* and is able to read and write NifTI-1 files as well as vtk files. In Figure 6.15, all software components and their connections are presented.

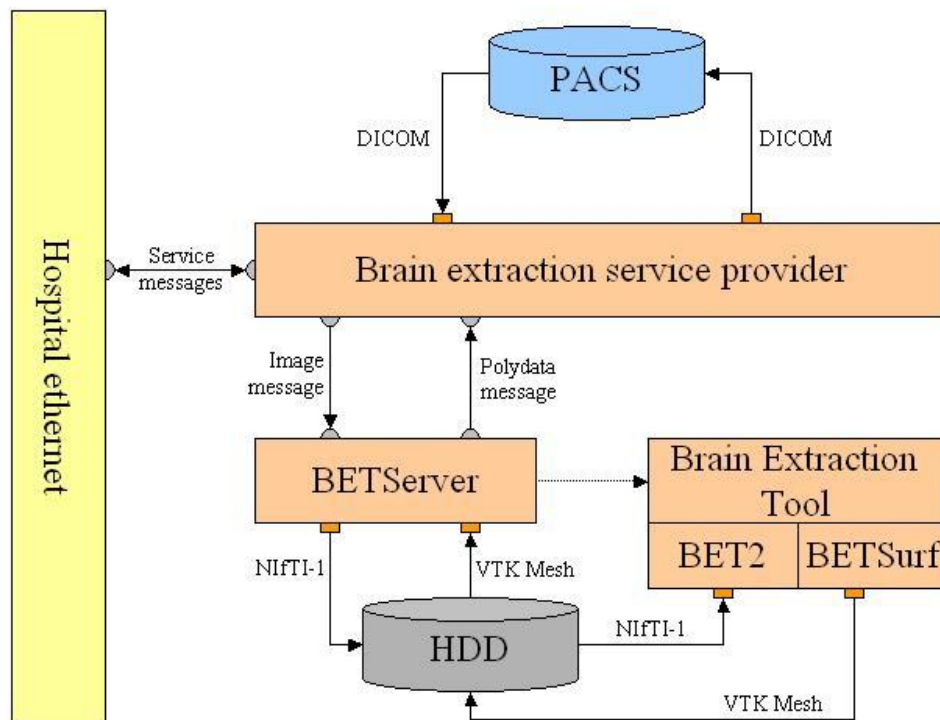


Figure 6.15: Collaboration diagram for preoperative cortex model generation.

The UML sequence diagram in Figure 6.16 gives an overview on the interaction between the modules:

- A scheduling workstation (*scheduler*) sends name and ID of the patients for whom segmentation is required to the *BESP* (1).
- The PACS server sends an `InstanceAvailable` message to the *BESP* using the DICOM service `N-EVENT-REPORT` (2, 3).

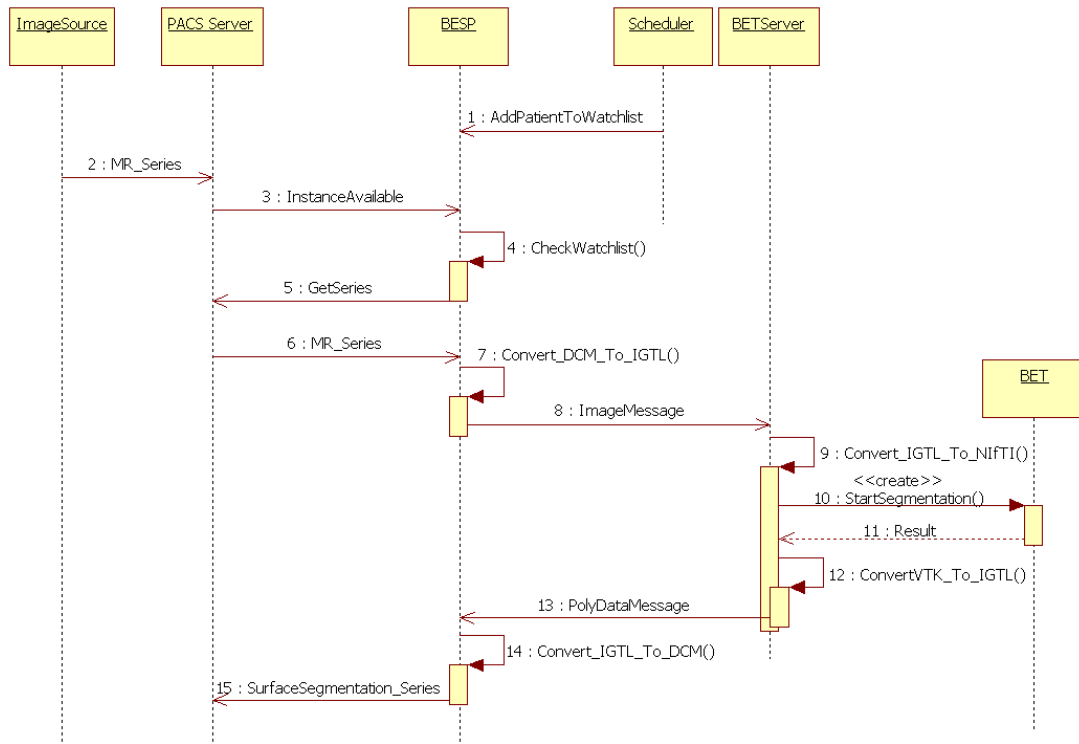


Figure 6.16: Preoperative cortex extraction: sequence diagram.

- The *BESP* checks whether the images belong to one of the patients it is waiting for and whether the instances are MR images. In this case, the *BESP* retrieves the instances from the PACS server (5, 6).
- The *BESP* sends the images to the *BETServer* as a TiCoLi *ImageMessage* (7, 8).
- The *BETServer* stores the images in the NifTI-1 format on the local hard disk and runs the *BET* (9, 10).
- After segmentation, the *BET* stores the resulting surface mesh on the local hard disk. The *BETServer* loads the surface and sends it to the *BESP* in a TiCoLi *PolyDataMessage* (11, 12, 13).
- The *BESP* generates a valid DICOM surface segmentation instance from the received surface mesh and the patient information in the original MR images. The surface segmentation instance is sent to the PACS server (14, 15).

Although it is running in a preoperative setup, the *BESP* is a prototype for a bridging device which connects a TiCoLi network with a PACS network. The *BESP* receives notifications and data from the PACS via DICOM services and translates the received data into TiCoLi messages which it forwards to a TiCoLi peer. Output data is received from TiCoLi peers, translated into DICOM instances and sent to the PACS server. This

pattern is one of many possible solutions to embed an intraoperatively used, distributed CAS system to into the PACS.

6.3 Summary and Discussion

Two applications were presented which utilize the presented methods for data exchange and system integration in clinically motivated projects. The first application uses the proposed DICOM SOP classes for exchange of surgical planning data in a pre- or perioperative planning scenario. The second application uses a combination of both, DICOM and TiCoLi messages to integrate an automatic segmentation tool, a scheduling workstation, and an intraoperative modeling system with the patient image database of the PACS. The intraoperative modeling system consists of several modules which interact in real-time using TiCoLi services. The throughput and latency of the TiCoLi transfer services was sufficient for intraoperatively using the modeling system.

In both applications, the surface segmentation IOD was used to exchange surface models between two applications. One limitation which constrained the level of automation in the exchange of data lay, in both cases, in the difficulty of the receiving application to identify the surface segmentation instance which to import. In situations where more than one instance was available for one patient on the PACS, the software required user input in order to select the correct one. To work around that problem, the segmentation systems in both setups use code names for segment labels by which the receiving applications were able to identify the correct surface segmentation instances. In order to solve this issue, a standardized mechanism for the identification of a DICOM instance which was generated during surgical planning and shall be used intraoperatively for a certain purpose is required.

For implantation planning results, such a mechanism is present in the implantation planning SR document. The instances of this definition are specific to one patient, one intervention type (identified through a standardized code), and one scheduled intervention date. A CAS system which has access to the OR schedule can automatically retrieve these instances for the next patient during OR turnover without the need for human interaction. The SR document contains references to all relevant data on the PACS.

The general applicability of DICOM SR for the representation of surgical planning results was noted by Treichel et al. [[Treichel et al., 2010](#)]. For future work items concerned with the storage and transfer of surgical planning results, the combination of IODs and SRs is recommended:

- The SR Document relates to one scheduled intervention. It contains references to patient images and models as well as to models of tools, devices, procedure steps, etc. and a coded identification of the planned procedure type.
- The models are represented by IOD instances which are defined independently from specific surgical interventions.

The seamless integration of TiCoLi application is also restricted by the absence of semantic standards. The service discovery mechanism of the TiCoLi in its actual version provides only the syntactical framework for autoconfiguration: with the service discovery mechanism, a technical description of the capabilities of a device can be exchanged. The device description contains a name and a rather unspecific type together with the network address at which the device can be contacted. The service-specific descriptions contain predominantly technical parameters, e.g. the framerate and frame size in the case of streams, together with a free-text name of the stream. In the presented systems, this mechanism did save a lot of setup effort: network addresses were generated and exchanged without any user interaction.

Nevertheless, the identification of a server required human interaction or a mutual agreement between two devices about their device names and service names. For fully automatic device discovery and *identification*, a mechanism for describing the semantics of a device or a service is required which is based on standardized codes. No such coding scheme exists which completely covers the concepts required to describe the domain of surgical device integration. Efforts are required to specify such a terminology. Thereby, existing coding schemes and standards from other fields of medical informatics might be adapted or extended for specification of codes on several levels of the device description:.

- For pathological concepts and laboratory findings, SNOMED and LOINC provide (almost) comprehensive terminologies which are continuously adapted to the needs of evolving technologies. For the labeling of biosignals in streams as well as in attribute descriptions and method parameter descriptions these terminologies could provide a starting point. Similar to the manner in which the DICOM standard refers to SNOMED and LOINC concepts for labeling of diagnostic findings, a coding scheme could be added to the service descriptions of the TiCoLi to identify biomedical content.
- For the description of image sources, a nomenclature for imaging modalities is required. The DICOM context group CID 29 ("Acquisition Modality") could be a good starting point for this nomenclature.
- Finally, a terminology is required for the identification of system parameters to which access is granted for remote monitoring and control as well as for operators, functions, and algorithms which are activated through the method service of the TiCoLi. The definition of a comprehensive and practicable descriptive scheme requires intensive research into the direction of an ontology for CAS functionalities, systems, and parameters. ISO 11073 (Point of Care Medical Device Communication) specifies a mechanism, a domain information model, and a nomenclature for the identification of device parameters in intensive care units. This standard could provide a good starting point for the development of a similar mechanism for surgical devices.

Chapter 7

Conclusion & Outlook

7.1 Summary

Pre- and Intraoperative Requirements

In Chapter 3, the functional requirements for inter-device communication in modular CAS systems were investigated. Surgical workflows were the starting point for the analysis of dataflow during preoperative preparation and planning as well as during intraoperative assistance. It appeared that during preoperative planning, versatile information which is stored in the HIS or its subsystems, predominantly the PACS and the patient health record, is utilized: an integrated planning system requires access to these systems through an adequate infrastructure for data exchange. This infrastructure is required to be accessible from each operating room in order to transfer preoperative data to intraoperatively used CAS systems. As it was pointed out in numerous publications (see Chapter 1), such an infrastructure is required to be based on open standards for data exchange and system integration. Only an infrastructure which facilitates inter-device communication across vendor boundaries delivers the required convergence of information, thereby giving rise to synergetic effects earned from the possibility to freely exchange input data, generated models, and planning results between systems. These effects help to make every-day clinical work more efficient and accurate.

The TIMMS meta architecture was cited as a potential reference model for such an architecture. TIMMS was specified with an intraoperative application in mind. It was argued in Chapter 3 that the differing requirements for pre- and intraoperative data exchange suggest the design of two separate information systems: a Surgical PACS (S-PACS) for preoperative planning (and postoperative follow-up and reporting) and the TIMMS for intraoperative system integration as a more interactive system with properties similar to an industrial field bus. An important feature to be kept in mind when designing both the S-PACS and the TIMMS is the interconnectivity between the domains.

S-DICOM

DICOM contains data structures for patient images as well as for textual or other de-

scriptive content which relates to image data. The majority of the information which was found in the dataflow in Chapter 3 is images or geometric data or related to either or both. In Chapter 4, DICOM is identified to be a suitable basis for a data storage and exchange standard for the perioperative data transfer. Nevertheless, the existing data structures which in DICOM do not fully cover all requirements which were identified for preoperative data exchange. Several data objects are listed in Chapter 4 for which DICOM does not contain applicable SOP classes.

Two new DICOM data structures were presented in Chapter 4. Both were defined in order to facilitate the storage and transfer of surgical planning data. The surface segmentation IOD (see Section 4.2) has already been adopted by the DICOM standard in 2008. The implant template and implantation planning IODs which were presented in Section 4.3 are as of January 2010 still being edited by the DICOM working groups 6 and 24. A final version of the supplement is planned to be released for voting before summer 2010.

Intraoperative Infrastructure

In contrast to preoperative planning workflows, the data exchange between intraoperatively used CAS systems is characterized by real-time transmissions and flexible setups. Three criteria were postulated according to which an OR network infrastructure has to be designed:

- The elicitation, transmission, and conflation of *in-situ* measurements (ranging from 1D signals to 3D volumetric images) in real-time and the combination of these with pre-operatively generated models are fundamental to the concept of MGS.
- The concept of OR integration or the so called *surgical cockpit* requires the possibility to monitor and control OR devices through virtual interfaces with which they exchange status information and control signals via the network infrastructure in the OR.
- In most hospitals, ORs are multi-purpose facilities in which different kinds of interventions are performed and which are often shared among surgical specialties. Only in rare cases will the configuration of devices and device settings which are required for two consecutive interventions be identical. OR turnover time is already a critical issue with regard to hospital efficiency. The increasing amount of technical devices utilized in modern ORs and the advance in connectivity between devices is expected to add to the complexity of OR setup procedures. A network infrastructure for integration of OR technology has to provide assistance during the configuration of an OR setup.

In Chapter 5, a software library was presented which has been developed to meet these requirements. The *TiCoLi* is based on open-source libraries and open standards. It provides a comprehensive set of functionalities according to the requirements postulated

in the final report of the OR 2020 workshop and other publications (see Section 2.2 and above) which it makes accessible through a uniform API.

In section 5.4 the results of a series of experiments was reported in which the performance and reliability of the presented infrastructure were investigated. No limitations could be identified to the throughput, latency, and reliability of the transfer services other than the limits which were inherited from the network infrastructure and operating system. Under conditions with high network traffic, the requirement for a network load management service was identified as a means to assure in-time delivery of stream frames.

A release of the TiCoLi into the public domain as an open-source software library is planned. This step aims at including designers, programmers, and users from other research centers than ICCAS into the development process of the library and to emphasize the utilization of the TiCoLi for module integration in research projects.

7.2 Conclusion

Researchers, clinicians, and industry agree about the potential that lies in the integration of surgery information systems, intraoperative assistance systems, and information systems used during post-operative follow up. Several international conventions recommended the design of a comprehensive information system which enables the fluent exchange of information between all departments and devices which directly or indirectly affect the perioperative workflow. This information system is expected to increase efficiency, outcome qualities and patient safety during all stages of patient care. The convergence of radiology information systems, picture archiving and communication systems, and computer assisted diagnosis systems which could be witnessed during the last decades demonstrates these effects.

The integration of the OR into the hospital-wide information systems for patient-data management is feasible. The DICOM standard is a viable basis for the exchange of planning data between the PACS and the OR. Additional DICOM data structures are required to fully cover the requirements of surgical planning, but the service classes and paradigms of the DICOM standard are consistent with these requirements.

The intraoperative data exchange via an Ethernet-based network is possible within reasonable constraints and technical boundaries regarding the reliability of transfer speeds. Using professional network hardware and when operating at bandwidths well below the limitations of the network, a high reliability of streaming data exchange could be measured. In a modern 1 *Gbit* network, most applications will adhere to this restriction. The feasibility of streaming under extreme conditions, such as very large bandwidth requirements for uncompressed stereo HD video or biosignals with high sampling rates above 100 *Hz*¹ was not sufficiently investigated. For use cases which

¹Electrocardiograms are, e.g., acquired at sampling rates of up to 8000 *Hz*. In order to stream this data through the TiCoLi, it would either have to be downsampled by, e.g. a sliding average filter, or sent in packets containing several hundred samples.

require such data transmission, the necessity may arise to complement the TiCoLi infrastructure with parallel communication infrastructures using dedicated Ethernet networks or other technology (such as analog or digital video routing). The modular software design of the TiCoLi facilitates the inclusion of interfaces to such infrastructures. The interaction with an additional infrastructure could be conducted by either adding a new class of services managed by a new `Manager` class or by adding new `Frame` or `Message` classes which are transmitted over specialized `Encoder` classes or a dedicated message socket.

Besides technical issues, such as the definition of data structures, the seamless integration of CAS systems into an information system requires a deep understanding of the semantics of the domain. In order to unambiguously describe the services of a device, an ontology is required which describes the tools, devices, and processes (technical as well as clinical) and their relations. Neumuth, Jannin, Raimbault, et al. [Jannin & Morandi, 2007; Neumuth *et al.*, 2009; Raimbault *et al.*, 2009] proposed methods for creating models of surgical procedures from observations and applying these models during surgical planning. Mudunuri et al. presented an ontological framework for the description of tools, procedure steps, personnel, and devices [Mudunuri *et al.*, 2007]. These and similar approaches have the potential to be the basis for the required ontology. In order to develop internationally accepted and expert-reviewed ontologies, an open database, a reviewing process, and a community which actively adds and reviews database content are required.

The legal, political, and social implications of the introduction of modular system architectures with shared functionalities into the surgical domain are manifold. The distribution of legal responsibility is at present an open issue. An open infrastructure allows for setups which include devices from different vendors. Whether and how such a setup can receive certification and clearance for clinical application on a patient is unclear. The scientific community which is promoting the development of integrated ORs must strike a balance between convincing legislators, clinical societies, as well as the general public and adjusting their aims to the basic conditions imposed by these groups.

7.3 Outlook

Computer assisted surgical planning is at present predominantly based on image data. On this basis, the storage of planning results in the PACS and their transfer into the OR with DICOM services is standing to reason given that the required data structures are added to the DICOM standard. DICOM Working Group 24 has been established for that purpose and has begun, with increasing support from the industry and other DICOM working groups, to bring DICOM supplements based on surgical use cases through the standardization process. Continuing effort will be required to proceed in that direction and to begin working on additional work items for other use cases in surgical planning.

The definition of the implant template IOD and the implantation planning SR document which will soon be brought to conclusion was only possible with the inclusion of several experts from the implant and implant planning industries. The extension of the work of DICOM Working Group 24 into other fields of surgical planning will require a comparable amount of input from the affected industries and clinicians.

Recent research projects in surgical planning go beyond image-based analysis of cases and include physical and physiological models into the decision process. These works require repositories for complex mathematical description of tissue behavior, biochemical, and biophysical processes. PACS may not be the optimal information system for handling this kind of information and DICOM may not be the optimal choice for a standard for storage and transfer of such data. The Physiome project has established a database scheme and a description language for physiological models. Surgical planning workstations of the future will have to be able to interact with such repositories, as well as with the patient data repositories in the HIS and PACS.

The formation of standardization bodies and the participation of the key players from healthcare, research, and medical device industry in these bodies are inevitable for the development and pervasion of technical standards for integrated ORs. For the perioperative storage and transfer of patient data, the DICOM standard and DICOM Working Group 24 provide a good basis in this regard. For intraoperative data exchange, no dedicated standard or standardization committee exists. Technical standardization bodies exist for certain aspects of intraoperative data exchange in CAS systems, such as the X3D consortium (which is maintaining an internet standard for exchange and visualization of 3D data) and the ISO Technical Committee 184/SC2 "Robots and Robotic Devices". The ISO has issued a call for experts in order to build a study group in medical care robots which will hold its first meeting in February, 2010. To improve the interaction between these and other standardization bodies and initiatives, efforts are currently made to start an IHE domain for surgery.

Appendix A

Data Flow Diagrams

Data Flow Diagrams (DFDs) [[Gane & Sarson, 1979](#)] are a modeling method in software engineering with which the exchange of information between the entities of a system is depicted. One DFD usually describes the flow of information which is exchanged to perform one action or within one use case. DFDs are directed graphs with three kinds of nodes:

- *External Entities* are objects outside the system which act as sources and/or destinations of the system's inputs and outputs.
- *Processes* take data as input and generate output data based on the input data.
- *Data storages* are entities which store information.

In principle, data flow diagrams are not restricted to digital data processing. Real-world processes where data exchange is done physically in the form of paper-based documents which are stored in filing cabinets can as well be modeled using data flow diagrams.

There exist different notations for DFDs. In this thesis, the notation of Gane and Sarson [[Gane & Sarson, 1979](#)] is used. It has the following elements:

- *External Entities* are represented by closed rectangles and are labeled with a speaking name.
- *Processes* are represented by rounded rectangles, are labeled with a speaking name, and contain a unique identifier (ID). The user or system who or which is involved in the process is named in the lower part of the node. DFDs are often modeled on several layers of abstraction where the internal structure of one process in a diagram is described by another DFD in more detail. To clarify the level of abstraction of a DFD, the process IDs in a sub-workflow contain references to their parent process: i.e., the sub processes of a process with the ID 1 would be identified by the IDs 1.1, 1.2,

- *Data storages* are depicted as open rectangles. They are labeled with speaking names and identified by IDs which usually consist of the capital letter 'D' and a unique number.
- *Data flow* between nodes is modeled as directed edges (arrows) between nodes which are labeled with speaking names to describe the content of the data which is exchanged along the edge.

The elements of the Gane and Sarson notation for DFDs are shown in Figure A.1. The figure contains the abstract definition of the elements of the notation and an example. The example depicts the way orders might be handled in a simple order management system:

- The customer hands in a filled in order form to a sales manager.
- The sales manager checks the order for consistency.
- Approved orders are send to the storage, unapproved orders are returned to the customer.
- Order processing is done by a packaging employee who takes orders from the order storage and sends the ordered goods to the customer.

The lower DFD in the figure is a fine-specification of process 1 in the upper diagram.

A.1 Diagram Types

DFDs were originally introduced as a means of system specification in software engineering. Following the DFD modeling process, for each system which is specified, five types of diagrams are required:

- A Context DFD where the whole system is represented on a very high level of abstraction as one single process connected via data flows with all external engines.
- Current physical DFDs capture the actual way an existing system is handling data to fulfill a purpose.
- Current logical DFDs are an abstraction of the current physical DFDs where the way data is managed in an existing system is described from a perspective which neglects the underlying technology.
- Proposed logical DFDs model the new design which is proposed to enhance the way, the system handles the data to fulfill the purpose. Again, the logical proposed DFDs are modeled from a technology-independent point of view.
- Proposed physical DFDs depict the way the new system handles data in a more detailed fashion which already contains details about the technology which shall be used to implement that data flow.

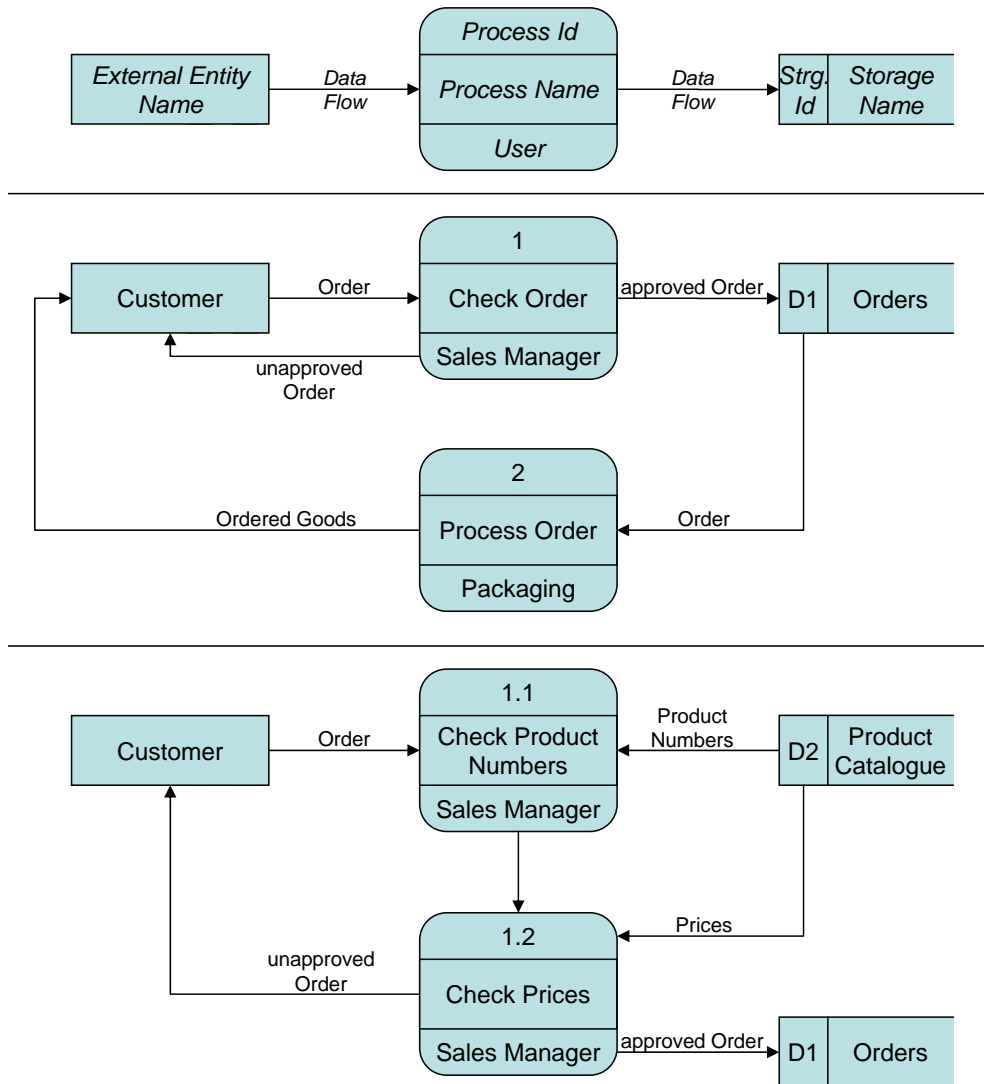


Figure A.1: Elements of the DFD notation of Gane and Sarson. Above: Abstract definition. Middle: Top Level Example. Below: Sub-DFD for process 1.0.

For the design of complex software systems, the DFD software design process has shown to be not flexible enough. On a high level of detail, the process of drawing and documenting DFDs will soon grow out of hand and it has been observed, that there is nearly no correlation between DFDs and the program code which is created in the implementation phase of software development. Nevertheless, on a high level of abstraction, when the overall handling of information between systems or subsystems needs to be visualized, DFDs still can be viable tools [Ambler, 2004]. In Chapter 3 of this thesis, DFDs are utilized to depict the information flow between TIMMS engines in planning situations and during intra-operative support to identify necessary connections and interfaces. For the fine-specification of the interfaces, more flexible and operable means of modeling are utilized, like UML class diagrams and UML interaction diagrams.

Appendix B

DICOM

B.1 The DICOM Information Model

DICOM specifies data structures, the so called *Information Object Definitions* (IODs), and services for handling of information objects. A service definition together with the specification of the information object to which the service applies is called a *Service Object Pair*, also called service object pair class (SOP class). SOP-Classes are the elements of DICOM data exchange: every activity performed using DICOM is the instantiation of one SOP class. From a database point of view, a SOP class Instance is a transaction. From a software engineering point of view, each SOP-Class implements one use case. The relation between services and objects is expressed in the fundamental DICOM information model (see Figure B.1). SOP classes are specified in part 4 of the DICOM standard. Each SOP class is identified by a globally unique identifier (UID).

B.2 DICOM Information Objects

Part 3 of the DICOM specification contains Information Object Definitions (IODs). An IOD is an object-oriented abstraction of information objects which represent real-world objects. IODs are standardized definitions that assure that different applications share a common view of the information objects they exchange. The term IOD is not an actual data set representing a specific entity of the real world, but rather a common specification about how data sets are to be constructed which represent real-world objects. In a sense, IODs are similar concepts as classes are in object-oriented terminology. DICOM distinguishes between *normalized IODs* which represent a single class of real-world objects, such as one patient, and *composite IODs* which represent a number of related classes, such as an image which is part of a study that has been acquired from one patient with a particular imaging device.

Inside IODs, DICOM attributes describe the properties of real-world entities. The

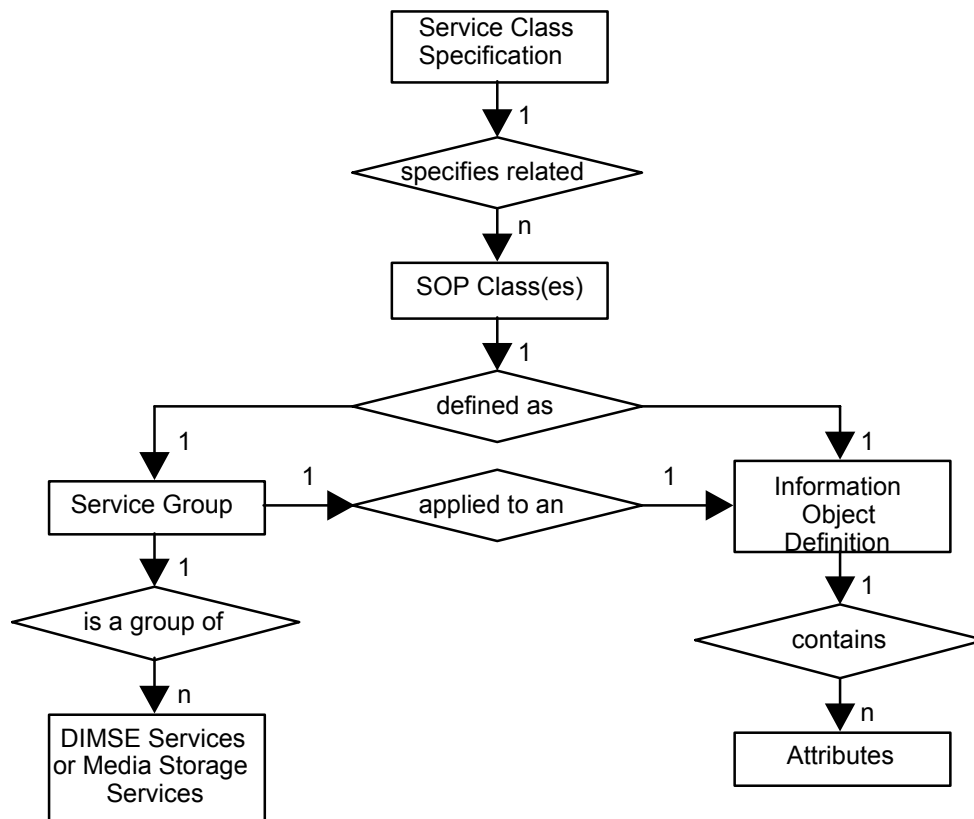


Figure B.1: Major structures of DICOM information model [NEMA, 2008b].

attributes of DICOM are specified in part 6 of the standard. For each attribute, DICOM specifies

- a unique identifier, called its tag, which consists of two 64-bit numbers written in brackets separated by commas (1234,5678),
- a data type, called Value Representation (VR),
- and the cardinality, called Value Multiplicity (VM), which can be any positive integer or a (not necessarily bounded) interval of non-negative integers.

The 28 different VRs of DICOM are specified in part 5 of the standard. DICOM contains VRs for signed and unsigned integer and floating point numbers of different lengths and precisions, character strings of different maximal lengths, but special VRs for certain purposes, like the person name, age string, or Unique ID VRs with very specific rules of representation and a specific semantic meaning. Part 5 of DICOM specifies rules for encoding attributes of each VR when storing DICOM instances.

DICOM IODs *aggregate* attributes to specify information objects for specific use

To facilitate readability and extendibility, the DICOM standard organizes the attributes of IODs in modules and macros which can be reused across the standard. Macros contain attributes and can contain other macros. Modules contain attributes and macros. IODs are combinations of modules (see Figure B.2).

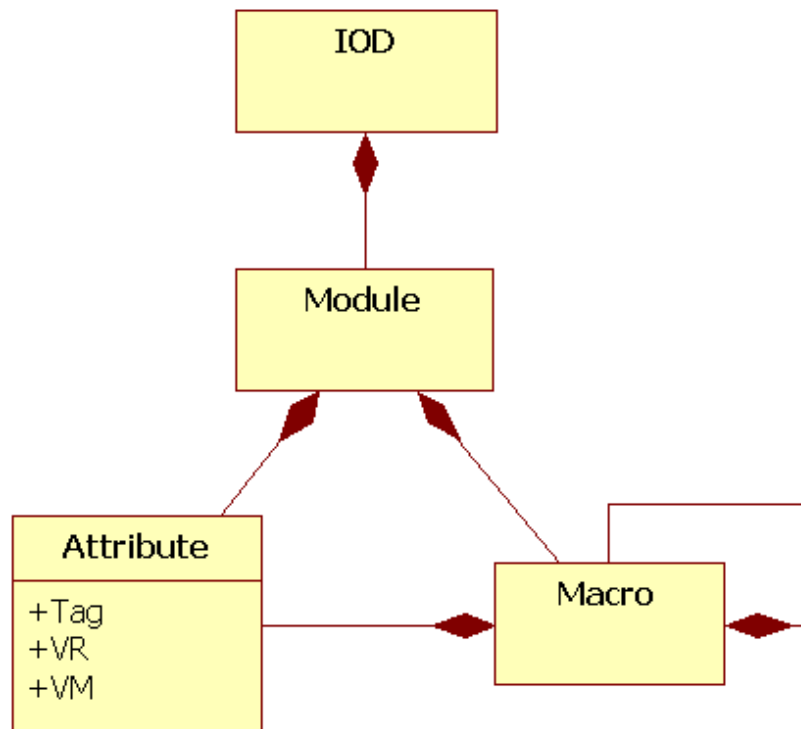


Figure B.2: Aggregation of attributes to IODs via modules and macros.

The content of DICOM Information Objects is based on a model of the real world. The model identifies the relevant real-world entities and their relations. The model which is depicted in Figure B.3 shows the model of the part of the world which is considered by DICOM for radiology workflows. DICOM contains separate world models for handling of print jobs and management of procedure steps. For a complete specification of these models, the reader is referred to the DICOM standard which can be obtained online from [NEMA, 2008b].

The DICOM information model is derived from the DICOM model of the real world. The entities of this model are IODs which represent the entities of the model of the real world. The model shown in Figure B.4 is modeled from a radiology point of

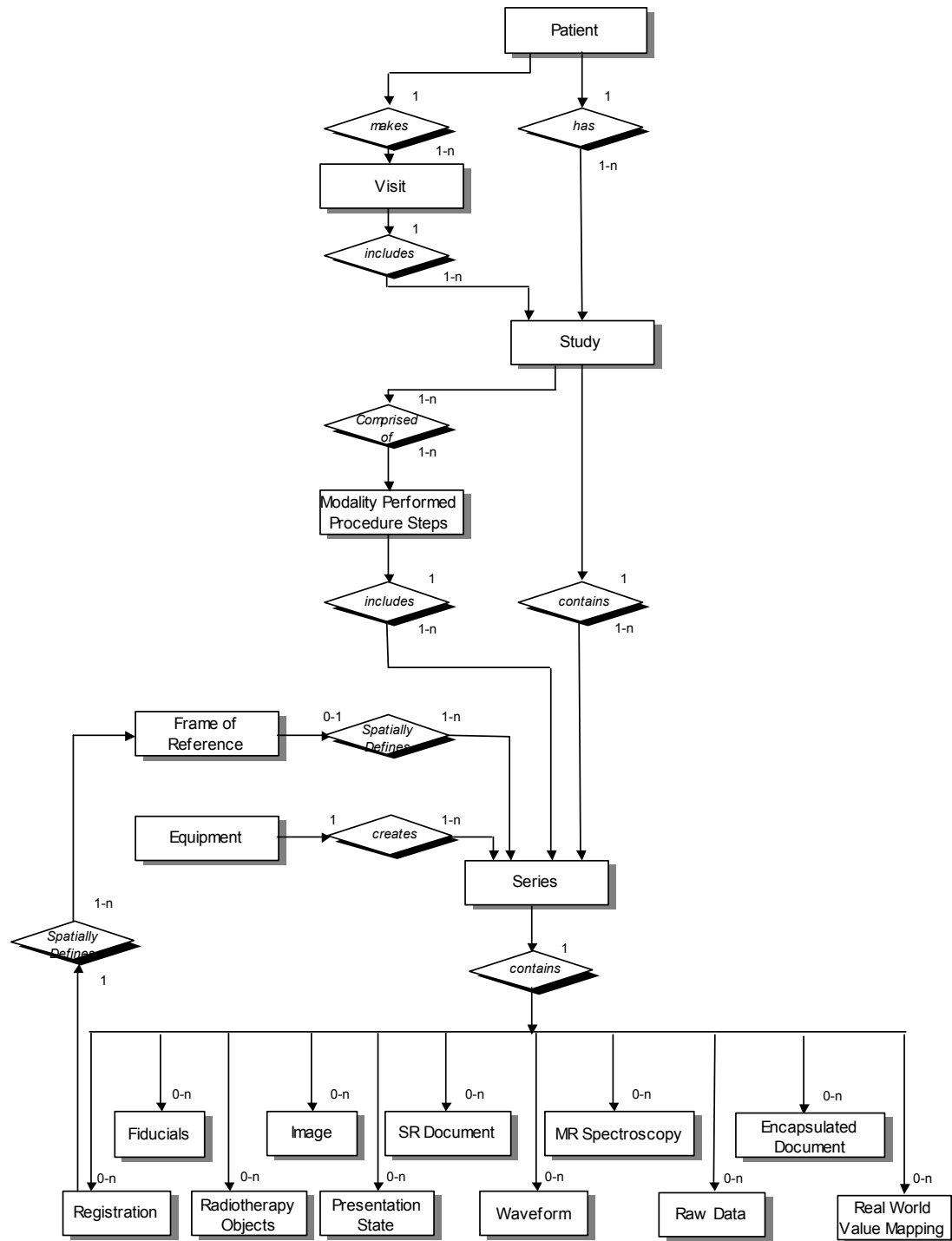


Figure B.3: DICOM model of the real world [NEMA, 2008b].

view. DICOM contains additional information models for radiotherapy IODs and other IODs, like the IOD which describes a requested hardcopy print job.

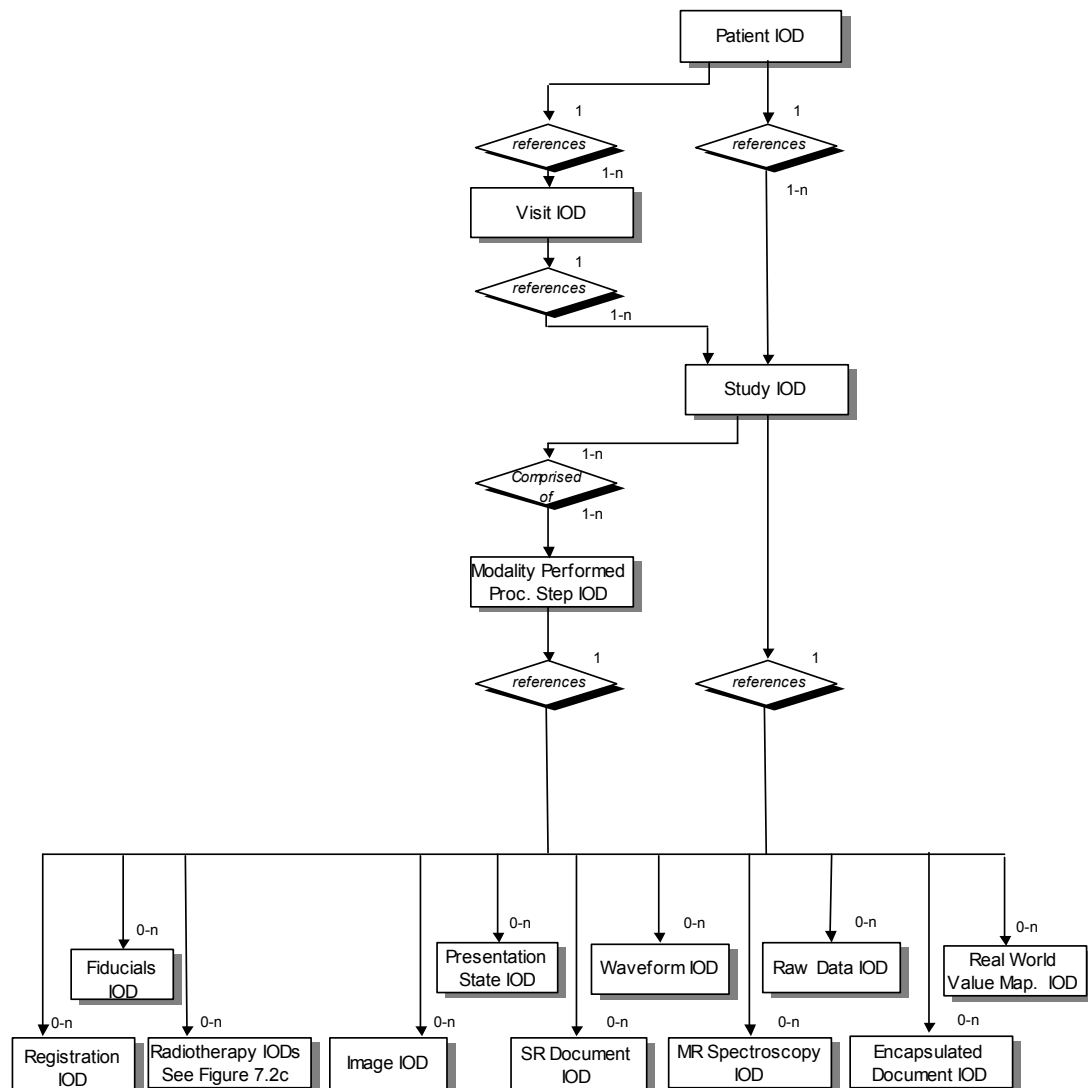


Figure B.4: DICOM information model [NEMA, 2008b].

B.3 DICOM Messages

Part 7 of the DICOM standard specifies the messages which are exchanged between DICOM Applications Entities (AEs), i.e. software systems which implement DICOM to communicate with other software systems. DICOM specifies a set of primitives, the so called DICOM Message Service Elements (DIMSE), from which complex services are composed. DICOM distinguishes between C-Services which apply to composite IODs and N-Services which apply to normalized IODs.

A DIMSE-service user is an application which is able to send and receive DICOM messages. For each service, there is an invoking and a performing DIMSE-service user. During one complex interaction, these roles can change between the applica-

tions several times: For each DIMSE service, the roles are distributed according to Figure B.5.

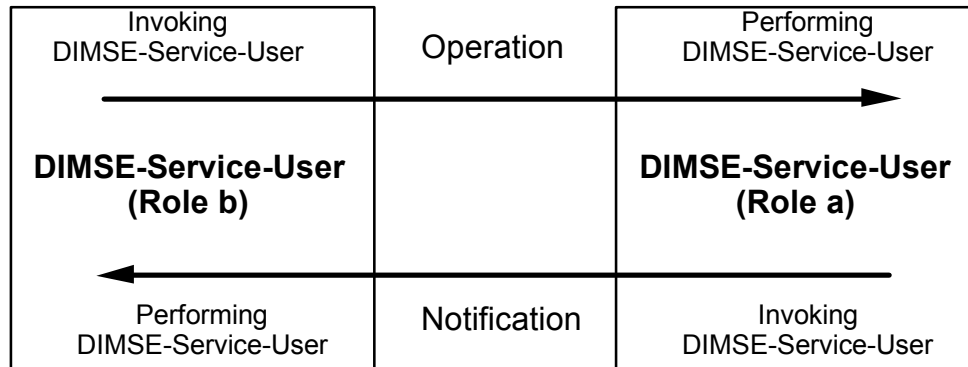


Figure B.5: DIMSE operation and notification flow [NEMA, 2008b].

DICOM services are acknowledged services. Each service invocation is handled using two-ways communication: The invoking DIMSE-service user sends a request-message to the performing DIMSE-service user. After performing the requested service, the performing DIMSE-service user sends a response message to the invoking DIMSE-service user. The generic sequence of DIMSE service primitives which are exchanged to negotiate one DIMSE service is shown in Figure B.6.

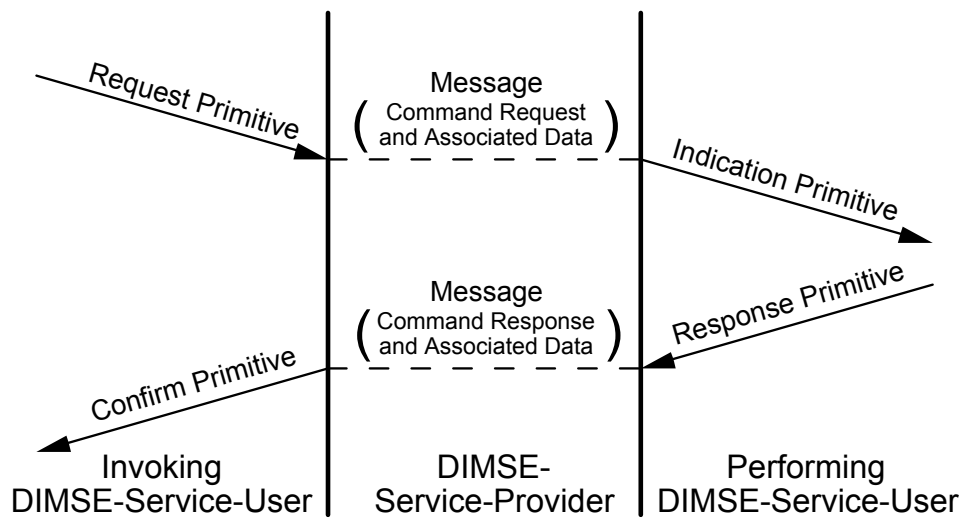


Figure B.6: DIMSE service primitives [NEMA, 2008b].

DIMSE Services

Part 7 of DICOM specifies the DIMSE services listed in Table B.1.

Name	Group	Type	Description
------	-------	------	-------------

C-STORE	DIMSE-C	operation	Invoked by DIMSE-service user to request the storage of a DICOM composite SOP instance at a peer DIMSE-service user.
C-GET	DIMSE-C	operation	Invoked by DIMSE-service user to fetch one or more composite SOP instances from peer DIMSE-service user.
C-MOVE	DIMSE-C	operation	Invoked by DIMSE-service user to request a peer DIMSE-service user to send one or more composite SOP instances to a third DIMSE-service user.
C-FIND	DIMSE-C	operation	Invoked by DIMSE-service user to match a set of attributes with the SOP instances managed by the performing DIMSE-service user. The result is a list of instances that comply with the query attributes.
C-ECHO	DIMSE-C	operation	Invoked to verify the end-to-end communication with the peer DIMSE-service user.
N-EVENT-REPORT	DIMSE-N	notification	Invoked by a DIMSE-service user to notify another DIMSE-service user about any event regarding a SOP-Instance. Like all DIMSE-services, this service is confirmed, i.e. a response is expected for every request which indicates that the request has been received.
N-GET	DIMSE-N	operation	Invoked by a DIMSE-service user to request information retrieval from a peer DIMSE-service user.
N-SET	DIMSE-N	operation	Invoked by a DIMSE-service user to request the modification of information by a peer DIMSE-service user.
N-ACTION	DIMSE-N	operation	Invoked by a DIMSE-service user to request execution of an action by a peer DIMSE-service user.

N-CREATE	DIMSE-N	operation	Invoked by a DIMSE-service user to request the creation of a SOP class instance by a peer DIMSE-service user.
N-DELETE	DIMSE-N	operation	Invoked by a DIMSE-service user to request the a peer DIMSE-service user to delete a SOP class instance.

Table B.1: DIMSE services.

B.4 DICOM Services

Part 4 of the DICOM Standard specifies service classes and specific services for the different IODs specified in part 3 of DICOM. Of the different service classes which are specified by DICOM, the applications discussed in this work utilize the following:

- **Storage:** The storage service class specifies an application-level mechanism for transfer of information objects. It allows one application to send (store) information objects to another application. Part 4 of the standard defines one storage service for every IOD specified in part 3.
- **Query/retrieve:** The Q/R service class specifies on application-level mechanism that allows one application to query another application for instances using attribute lists as query keys. For each IOD, a query/retrieve service is specified. The specifications include the attributes one can query for. The Query/retrieve services make use of the C_FIND, C_GET, and C_MOVE operations.
- **Structured Reporting Storage:** An extension of the standard storage service class which extends the capabilities of the SCU and SCP toward exchanging structured report instances.
- **Instance Availability Notification:** The instance availability notification service class is utilized to send updates on the availability of instances to peer applications. The calling application calls the N-Create operation to send the notification to the receiver of the message.

Appendix C

TiCoLi Protocols and Libraries

OpenIGTLink

The OpenIGTLink protocol defines message-based data exchange. The protocol and an implementation of the protocol can be obtained from <http://www.na-mic.org/Wiki/index.php/OpenIGTLink>.

OpenIGTLink enables the exchange of messages from one application through an TCP/IP based network to another application. The object-oriented implementation distinguishes between different types of messages to facilitate interpretation of incoming messages. The exchange of messages is not based on a connection handshake, session initialization, or other prior - any application can send a message at any time to every peer of which it knows the TCP/IP socket address. Each message which is sent is independent of all other messages which have been exchanged or will be exchanged between these or other applications in the network.

A universal message format is specified: Each message consists of a 58 bytes long header and a body of arbitrary length. The structure of an OpenIGTLink message and the internal structure of the message header are shown in Figure C.1. The content of the header of an OpenIGTLink message is described in Table C.1.

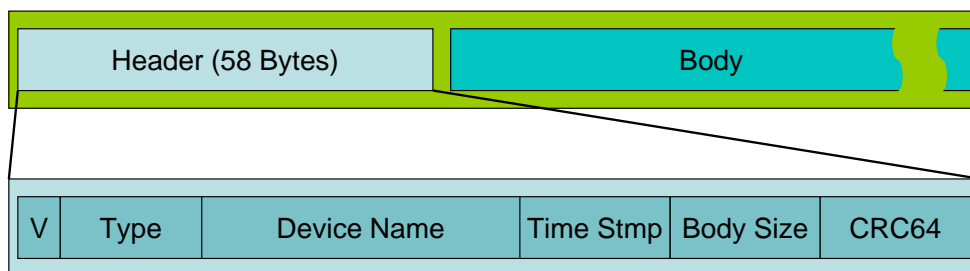


Figure C.1: Structure of an OpenIGTLink message.

The body of an OpenIGTLink message can either be a command or data. In principle, the length of the Body can be anything between zero and whatever the underlying hardware and software are able to address. OpenIGTLink specifies seven commands, but allows extension of the protocol by additional commands. Data messages can contain any

Field	Bytes	Type	Description
V	2	Unsigned Short	Protocol Version Number
TYPE	12	Character String	Name of body data type
DEVICE_NAME	20	Character String	Unique name of the sending device
TIME_STAMP	8	Unsigned Integer	Message time stamp
BODY_SIZE	8	Unsigned Integer	Length of message body in bytes
CRC	8	Unsigned Integer	64 bit CRC checksum for message body

Table C.1: Format of an OpenIGTLink message header.

of the standard data types for which OpenIGTLink specifies encodings or any other byte stream the sender and receiver of the message have a common understanding for that goes beyond the OpenIGTLink specifications. Since OpenIGTLink is an open source project, users of the library are intended to extend both, data and command specifications for their applications and commit their additions to the project.

RTP

The Real-Time Transport Protocol (RTP) was introduced in 1996 by RFC 1889 as an internet standard for continuous transmission of audio and video data streams for end-to-end communication. In 2003, the original specification was replaced by RFC 3550. RTP is an application-layer protocol which facilitates packet-based data exchange via the User Datagram Protocol (UDP) transport layer. RTP consists of two components, the Data Transfer Protocol and the RTP Control Protocol (RTCP) which enables the exchange of reception quality feedback and synchronization messages. Since the underlying transport layer, UDP, does not have any session management, RTP requires session management on the application layer. Before an RTP session can be initialized, the sender and receiver need to negotiate the session parameters. RTP does not specify how this negotiation shall be done. Several protocols can be utilized for that purpose, among which the Session Description Protocol (SDP) and the Real Time Streaming Protocol (RTSP) are the most common. In the TiCoLi, an implementation of the RTP protocol which was published by Jori Liesenborgs is used. The package can be obtained from <http://research.edm.uhasselt.be/~jori/page/index.php?n=CS.Jrtplib>.

Service Discovery and Auto Configuration

The ZeroConf protocol facilitates the automatic configuration of an IP-based network without they need central network management entities, such as DHCP and DNS servers. ZeroConf specifies solutions to three major issues one needs to deal with in order to establish a computer network based on the IP:

1. **Automatic Address Assignment:** In order to identify systems in a network, the need to select or be assigned unique addresses. Methods for the automatic selection of IP addresses are part of the specification of the Internet Protocol in both versions, IP4 as well as IP6. There exists a special address space which was dedicated for automatically assigned addresses by the Internet Assigned Numbers Authority (IANA). In principle, the link-local addressing for IPv4 networks method goes as follows: A computer selects an arbitrary IP address in the address block between 169.254.1.0 and 169.254.254.255 and tests, whether the address has already been registered by another computer. If a collision is detected, the system has to try other addresses until it finds a vacant one.
2. **Name resolution:** In the absence of a Domain Name System (DNS) Server, the participating computers in a network need to create, update and distribute a list which maps hostnames and addresses by themselves. The idea behind the multicast DNS (mDNS) protocol is that every computer holds its own list of DNS records. When a system needs to resolve a hostname it does not have a record for, it sends a DNS request via a special UDP socket (224.0.0.251:5353) to all computers which are connected to the network. In order to keep the network traffic which originates from this service as low as possible, mDNS specifies how multicast-DNS requests which are sent to the computers in one network segment are to be handled and how clients and servers shall behave to reduce the number of redundantly sent requests and replies.
3. **Service Discovery:** Computer networks are usually built for the purpose of data exchange between systems and for making available functionalities of one system for other systems in the network. One prominent example for the latter are network printers to which every computer in a network can send print-jobs. Service discovery is a mechanism where devices in a network exchange information about the services they are offering. The DNS-Service Discovery (DNS-SD) protocol specifies how DNS service messages are used to describe the services a system is offering. The description contains a service type which is to be selected from a normative list. Each service type is associated with one protocol.

The TiCoLi uses the BonjourSDK implementation which is an implementation of the zeroConf specification by Apple Computer Inc. The BonjourSDK can be obtained from <http://developer.apple.com/networking/bonjour/download/>.

Network Time Protocol

The Network Time Protocol (NTP) is an application layer protocol which facilitates synchronization of the system clocks of devices in a network. It is designed to compensate for network latencies which are a common problem in packet-switched domains, such as IP based networks.

NTP is based on a hierarchical client-server architecture with highly reliable clocks (such as atomic clocks) on the highest level which is also called *Stratum 0*. On the first

level, Stratum 1, there are computers which are directly attached to Stratum 0 clocks. Stratum 1 systems are often called time servers. Systems in lower levels (Stratum n , $n \in [2, 255]$), send NTP requests to Stratum $n - 1$ systems. Usually, a Stratum n system send requests to more than one Stratum $n - 1$ system and interact with other Stratum n systems. NTP uses a modified form of Marzullo's Algorithm [[Marzullo, 1984](#)] to create a statistically optimal estimation of the system time from the answers it receives from its peer Stratum n and $n - 1$ systems.

Inside the TiCoLi, the system time of all components needs to be synchronized. NTP is used to ensure synchronization between the devices in one setup. It is proposed that one system in the OR acts as time server for the other devices. An example for such a setup is given in [[Bohn et al., 2009](#)].

Appendix D

Algorithms and Implementation Details

D.1 HandleSets

The `HandleSet` class is a container class used internally by the TiCoLi Core and Managers to store information about local and peer services. In principle, the `HandleSet` is a vector container which identifies each element of the vector with a hashed integer `Handle`. The `Handle` uniquely identifies one element within the container. `HandleSet` has the following public methods:

- `Allocate(unsigned int size)` empties the container and reserves `size` number of fields.
- `Handle Add(T &item, Handle handle, bool replace)` adds an item to the container. If `handle==0`, a random handle is assigned. The container is resized if no free field is found. If `handle!=0`, this handle is assigned to the item when it is added. If an item exists with the same handle, it is replaced when `replace==true`. In all cases, the `Handle` under which the item is stored in the container is returned, if adding was successful. Zero is returned otherwise.
- `bool GetItem(Handle handle, T &item)` assigns the item stored under `handle` to `item`. Returns `false` if `handle` cannot be resolved.
- `bool HasItem(Handle handle)` returns `true` if an item exists in the container with the given `Handle` and `false` otherwise.
- `Handle FindFirst(T &item)` Returns the `Handle` of the first item in the container and assigns this item to `item`.
- `bool Remove(Handle handle)` removes the item with given `Handle` from the container if such an item exists. Returns `false` if `handle` cannot be resolved.

- `void RemoveAll()` The container is emptied.
- `unsigned int GetSize()` The size of the container is returned. Since the container can contain empty fields, this number is not to be misinterpreted as the number of contained items.
- `unsigned int GetNumberOfItems()` The number of used fields in the container is returned. This number can be less than the size of the container.
- `HandleSetIt InitTraversal()` Creates an iterator with which the container can be accessed sequentially. The iterator points to first item in the container after creation.
- `bool GetNextItem(T &item, HandleSetIt &it)` Returns the item the iterator `it` currently points at and moves the iterator to the next item or to an invalid item if no next item exists in the container. Returns `false` if called with an iterator that points to an invalid item.
- `Handle GetCurrentHandle(HandleSetIt &it)` Returns the Handle of the item the iterator `it` currently points at.

D.2 Thread Safe Callbacks

In Listings D.1 – D.3, the process of thread safe callback invocation is presented as c++ code. The sequence diagram in Figure D.1 shows how the methods interact on the example of a message callback.

- The application, represented by `anApplicationClass` creates an instance `messageCallback<appClass, Message*>` which points to its message handling method and sets the `MessageManager` of the `TiCoLi` (see below) to sending incoming messages to this function (1,2).
- In the event of an incoming message, the `MessageManager` calls `Call()` on `messageCallback` (3,4).
- `messageCallback::Call()` calls `startCallback()` to check, whether not too many threads already are currently operating on this callback. The return value `OK` indicates that the counter could be successfully incremented (5,6).
- A thread is created which takes care of method execution. `Call()` is finished and returns `OK` to the `MessageManager` which can from this point on proceed with what ever he was doing besides handling this incoming message (7,8). All following steps happen within `cbThread`

- `cbThread` calls `messageCallback::startCallbackThread(\neg)` which calls `CallFromThread()` from where the member function of `anApplicationClass` is actually called. (9,10,11).
- After the function is executed, `CallFromThread()` ends and `startCallbackThread()` calls `StopCallback()` to decrement the thread counter before the thread is destroyed (12,13,14).

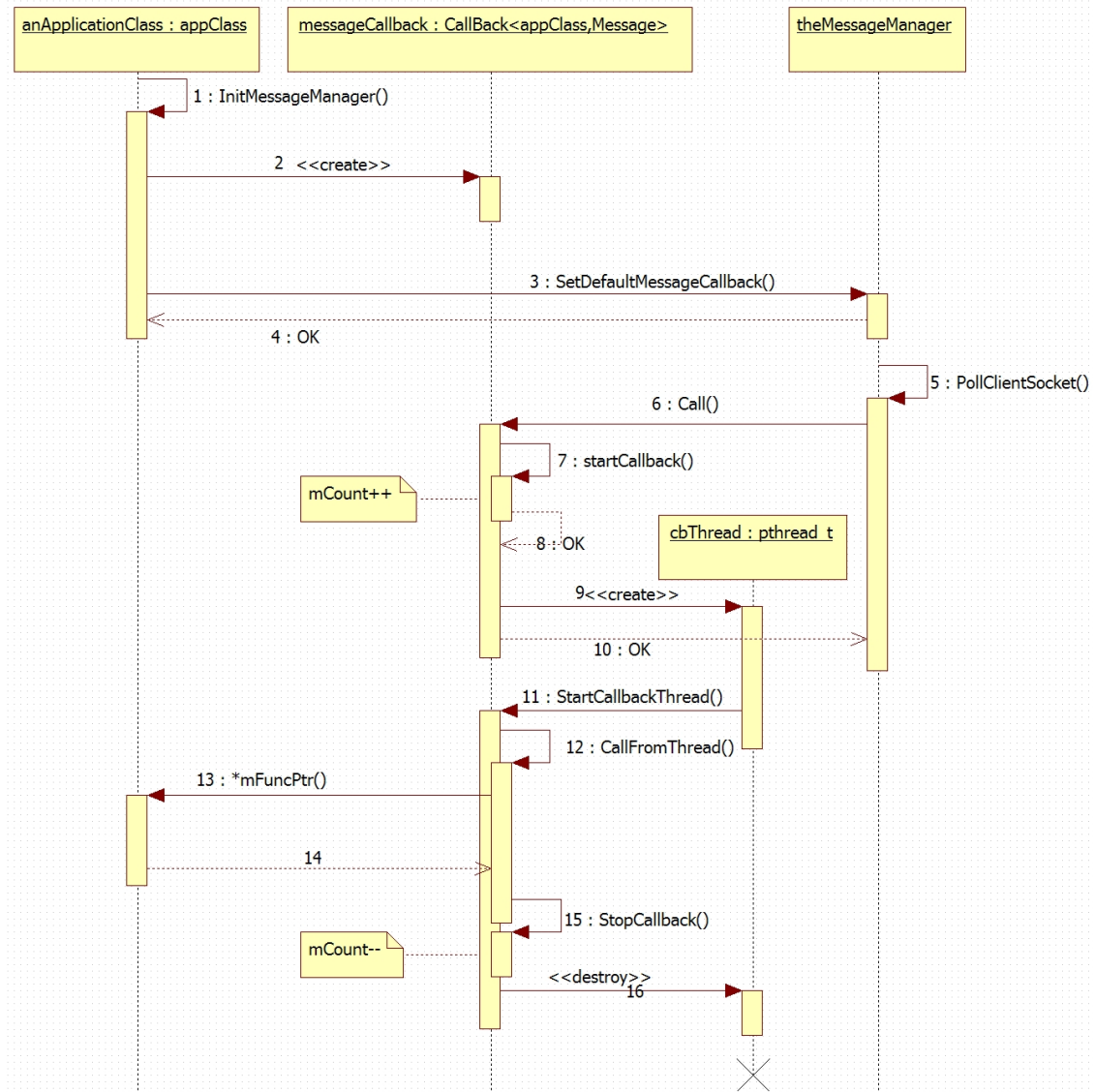


Figure D.1: UML sequence diagram for TiCoLi callback execution.


```

1 // A construct required for handing the required data into a
  // thread which executes a callback
2 template <class P>
3 class CallbackThreadData
4 {
5 public:
6     CallbackThreadData(P v, Handle h, void *t)
7         :value(v)
8     {
9         handle=h;
10        This=t;
11    }
12
13    P value;
14    Handle handle;
15    void *This;
16 };
17
18 //Counts the number of threads which run one callback method.
19 class CallbackThreadCounter
20 {
21 protected:
22     static pthread_mutex_t mCountMutex;
23     static unsigned char mCount;
24     static unsigned char mCountMax;
25     //Try to increment the counter
26     static TiCoLi::Condition StartCallback()
27     {
28         Condition result=BUSY;
29         pthread_mutex_lock(&mCountMutex);
30         if(mCount<mCountMax)
31         {
32             mCount++;
33             result=OK;
34         }
35         pthread_mutex_unlock(&mCountMutex);
36         return result;
37     }
38     //Decrement the counter
39     static void StopCallback()
40     {
41         pthread_mutex_lock(&mCountMutex);
42         mCount--;
43         pthread_mutex_unlock(&mCountMutex);
44     }
45 };

```

Listing D.1: Internal Class Specifications for Callback Handling. Constructors and destructors are omitted.

```

1 //Abstract Interface to CallbackMethods
2 template<class P>
3 class CallbackBase : public CallbackThreadCounter
4 {
5 public:
6     virtual Condition Call(P value,Handle handle)
7     {
8         if(IsNull())
9             return DOESNT_EXIST;
10        Condition c=StartCallback(); //try to increment thread
            counter
11        if(c!=OK)
12            return c;
13        pthread_mutex_lock(&mCallMutex); //gets unlocked in
            CalledFromThread()
14        //create the thread and return OK
15        pthread_t cbThread;
16        CallbackThreadData<P> *threadData=new CallbackThreadData<P>
            >(value,handle,this);
17        pthread_create(&cbThread,NULL,&CallbackBase<P>::
            startCallbackThread,threadData);
18        return OK;
19    }
20
21    virtual bool IsNull() = 0;
22    virtual CallbackBase *SpawnCopy() = 0;
23
24 protected:
25     //called from the thread to execute the method
26     static void* startCallbackThread(void *data)
27     {
28         CallbackThreadData<P> *tData=(CallbackThreadData<P>*)data;
29         CallbackBase<P> *This=(CallbackBase<P>*)tData->This;
30         This->CallFromThread(tData->value,tData->handle);
31         //After the function is executed, the thread returns here.
32         delete tData;
33         StopCallback(); //decrement thread counter
34         return NULL;
35     }
36
37     virtual void CallFromThread(P value,Handle handle) = 0;
38
39     pthread_mutex_t mCallMutex;
40 };

```

Listing D.2: CallbackBase class. Constructors and destructors are omitted.

```

1  template <class T, class P>
2  class Callback : public CallbackBase<P>
3  {
4  public:
5      virtual bool IsNull()
6      {
7          return (mFuncPtr==NULL) || (mInstance==NULL);
8      }
9
10 protected:
11     //This calls the function pointer.
12     void CallFromThread(P value,Handle handle)
13     {
14         pthread_mutex_unlock(&mCallMutex);
15         mInstance->*mFuncPtr(value,handle);
16     }
17
18 protected:
19     void (T::*mFuncPtr) (P,Handle); //Pointer to a function
20     T* mInstance;                  //Pointer to an instance of T
21 };
22 
```

Listing D.3: Callback class. Constructors and destructors are omitted.

D.3 The Gesture Detection Module

Figure D.2 shows the architecture of the gesture detection module. It contains a ring buffer of configurable length and seven components. The *Gesture Detection Job Control* module is the central controlling entity which instantiates and controls the other modules. For each tracked object the software is monitoring, it creates one *Detection Job*, which creates instances of five modules:

- A *Ring Buffer* buffer for short-term storage of received tracking coordinates,
- a *Coordinates Listener* which acts a TiCoLi streaming client to receive the coordinates and store them in the buffer,
- one module for the detection of each of the two gestures,
- and the *Result Message Sender* which sends notifications to all registered clients whenever a gesture was detected.

The coordinates listener, result message sender, as well as the point and loop detection modules are implemented to run in separate threads which access the ring buffer as a shared memory interface.

Ring Buffer

The ring buffer is the central data structure through which the gesture detection classes get access to the received tracking frames. The buffer has a fixed length and is accessed through iterators which traverse the buffer step by step. The iterators are implemented cyclically: when one iterator reaches the end of the buffer, it is reset to the beginning of the buffer. This results in a ring-topology where the elements of the buffer are written and re-written periodically.

Semaphores ensure that the coordinate listener is not getting write-access to fields as long as a read-iterator is reading data from that field. Multiple read iterators are given access to one field.

Coordinates Listener

The coordinates listener is a TiCoLi streaming client receiving tracking coordinates from any tracking module attached to the TiCoLi infrastructure. It is created and initialized by the detection job which provides it with the name of the tracking device and the stream offered by this device to which the listener connects. Once the connection is established, the stream listener will write all tracking data it receives to the

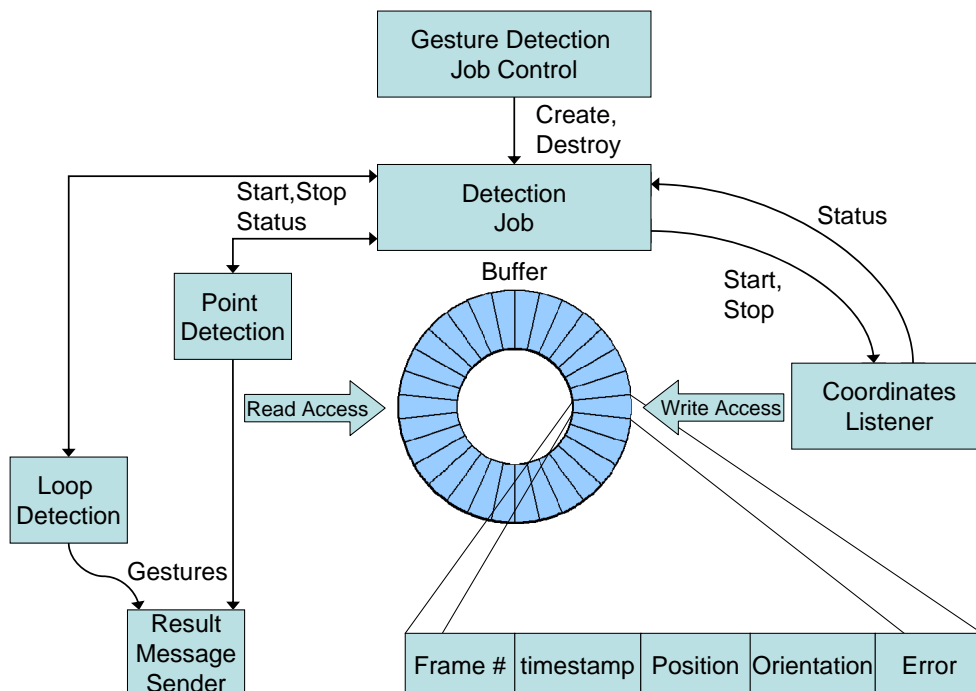


Figure D.2: Architecture of the gesture detection module.

ring buffer. The stream listener only frames which are labeled `visible`, i.e. frames which contain valid tracking information. Invalid frames are skipped.

Point Gesture Detection

The point detection thread searches the buffer for periods with minimal changes in the position. The point detection module is a state machine with two states (see Figure D.3):

- The initial state waits for a situation, where $N \in \mathbb{N}$ consecutive positions are within a small radius r (" $\Delta p < r$ "). In that event, the machine switches to the second state.
- The second state is left, when an input point is found with (" $\Delta p > r$ ").

The distances Δ are calculated according to a weighted mean $\tilde{\mathbf{p}}_i$ which is calculated from consecutive points which do not break the and its standard error $\tilde{\sigma}_i$ are initialized with the position and standard deviation of the first incoming point. Both are adjusted from incoming points \mathbf{p}_i and their standard deviations σ_i in a simplified Kalman approach (see [Kalman, 1960]), where the x, y , and z coordinates are treated separately and no system-inherent motion is modeled. Listing D.4 shows the point detection algorithm.

The loop (lines 24 –44) in `DetectPoint` continuously reads the buffer. All new measurements are processed in the Kalman filter implementation `Kalman(...)`. There, the distance between the new point and the actual estimate of the mean is calculated (line 6) and compared with a global cutoff-distance (line 7). If the new point is close enough to its predecessors, the innovation covariance S and optimal Kalman gain K are calculated and `mean` and `sigma` are updated according to Kalman's theorem. The `Kalman(...)` method returns `true`, if the new point was close to the mean and `false` otherwise.

The state machine from Figure D.3 is implemented in the loop in `DetectPoint` by altering the flag `isWaiting` and the counter `foundSimilar`. In the waiting state, `foundSimilar` is incremented whenever a point `newPoint` is processed and `Kalman(mean, sigma, newPoint, newSigma) == true`, i.e. if the new point is close to the actual mean estimation. Otherwise, the counter will be reset to zero and the model is reinitialized with `newPoint` and `newSigma` as new estimation. A point gesture is assumed, when N consecutive points are lying within the `cutoffdistance` around their mean. In that event, the state machine performs the transition into the second state, in which it waits for the end of the point gesture, i.e. until a point is processed which is further away from the actual mean than the `cutoffdistance`. In that event, the last estimated mean is output and the machine returns to the initial state where it waits for a new point gesture to begin.

Loop Gesture Detection

The loop detection thread implements the search for closed (or almost closed) loops

```

1  bool Kalman(double* mean, double* sigma, double* nextPoint,
    double* nextSigma, double cutOffDist) {
2      double y[3]; //Innovation
3      y[0] = nextPoint[0] - mean[0];
4      y[1] = nextPoint[1] - mean[1];
5      y[2] = nextPoint[2] - mean[2];
6      double dist = sqrt(y[0]*y[0]+y[1]*y[1]+y[2]*y[2]);
7      if (dist < cutOffDist) { //Point is close to predecessor(s)
8          for (int i = 0; i < 3; i++) {
9              //Calculate Innovation Covariance and Optimal Kalman Gain
10             double S = sigma[i] + nextSigma;
11             double K = sigma[i] / S;
12             //Update mean an sigma
13             mean = mean + K*Y;
14             sigma = (1 - K) * sigma;
15         }
16         return true;
17     }
18     return false;
19
20 DetectPoint(double cutoff) {
21     bool isWaiting = true;
22     int foundSimilar = 0;
23     double mean[3] = {0,0,0};
24     double sigma[3] = {FLOAT_MAX,FLOAT_MAX,FLOAT_MAX};
25     double newPoint[3],newSigma[3];
26     while(!mStop) {
27         if (newPointAvailable(newPoint,newSigma)) {
28             if (isWaiting) {
29                 if (Kalman(mean,sigma,newPoint,newSigma,cutoff))
30                     foundsimilar++;
31                 else {
32                     foundsimilar = 0;
33                     mean = newMean;
34                     sigma = newSigma;
35                 }
36                 if (foundsimilar > N)
37                     isWaiting = false;
38             } else {
39                 if (!Kalman(mean,sigma,newPoint,newSigma)) {
40                     SendPointGestureToReceivers(mean,sigma);
41                     foundsimilar = 0;
42                     isWaiting = true;
43                     mean = newPoint;
44                     sigma = newSigma;
45                 } else
46                     sched_yield();
47             }
48         }
49     }
50 }

```

Listing D.4: Point Gesture Detection.

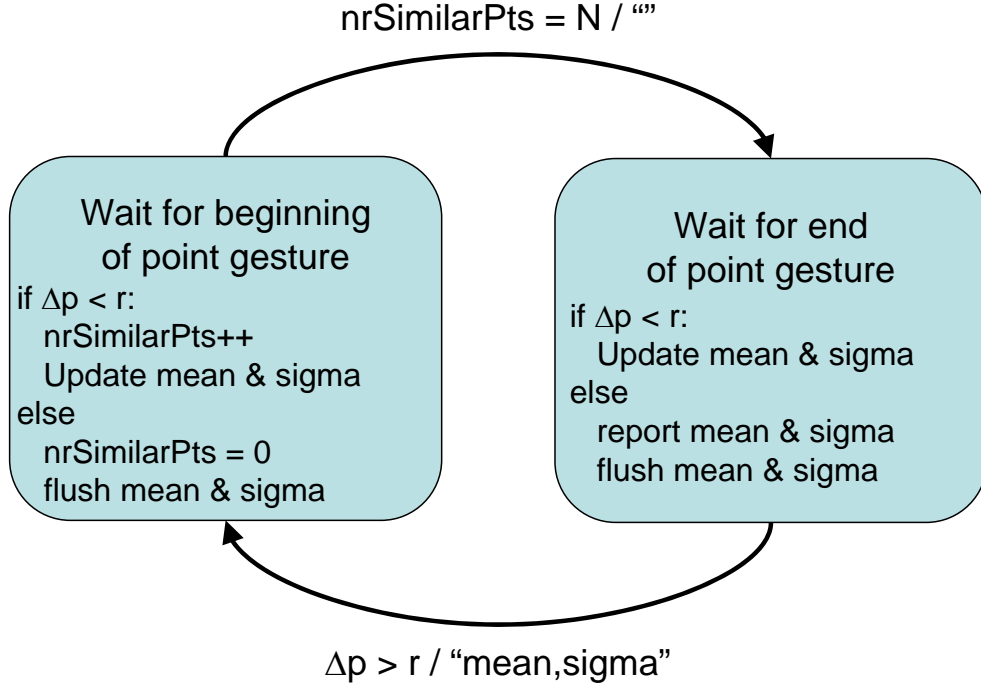


Figure D.3: Point gesture detection state machine.

in the received data. Loops are sequences of points which self-intersect or nearly self-intersect after showing some relevant motion. I.e., sequences $P = (\mathbf{p}_0, \dots, \mathbf{p}_n)$ where $\mathbf{p}_0 \approx \mathbf{p}_n$ and $\exists \mathbf{p}_i \in P : \|\mathbf{p}_0, \mathbf{p}_i\| \gg 0$. The first criterion describes the self-intersection property of loops, the second criterion ensures, that point gestures are not misinterpreted as loops.

The detection of self-intersections is a Nearest-Neighbor-Search (NNS) problem. NNS problems are defined as follows:

Given a set of n points $P = \{\mathbf{p}_0 \dots \mathbf{p}_{n-1}\} \subset \mathbb{R}^3$, a metric $L(\mathbf{p}_1, \mathbf{p}_2) \rightarrow l \in \mathbb{R}$, and a query point \mathbf{q} , identify the index i of the point \mathbf{p}_i with:
 $L(\mathbf{q}, \mathbf{p}_i) \leq L(\mathbf{q}, \mathbf{p}_j) \forall j \in [0; n - 1]$.

There exist variants of this definition for problems, where all neighbors which lie within a certain radius, or a constant number k of points which are the closest to the query point (k-NNS problems) are to be found. Other variants limit the search for the nearest neighbor to a cutoff distance. Donald E. Knuth gives a comprehensive overview on NNS problems in [Knuth, 1998].

Evidentially, the algorithmic complexity of NNS problems is $\Omega(n)$ or $O(\log n)$ if preprocessing of the points P is allowed. Despite this theoretical *worst case* boundary, algorithms have been developed which solve the problem *at average* with constant ($O(1)$) effort, i.e. in other scenarios than the worst case, such algorithms perform much better than the theoretical boundary. Algorithms based on the *cell technique* [Bentley & Friedman, 1979] have been proved to solve NNS Problems in constant

time for point sets which are equally distributed [Bentley *et al.*, 1980]. In contrast to other algorithms (for instance algorithms which are based on subdividing trees), the preprocessing effort for bucket search algorithms is linear and the algorithms can be applied in dynamic scenarios, i.e. with continuously updated point clouds. Bucket search algorithms divide the search-space into a regular array of buckets $B = \{B_j\}$ which form a tessellation of the search space. The tessellation is chosen so that a function $t(\mathbf{p}) \rightarrow \mathbb{N}$ exists which assigns a point to a bucket. Tessellations are chosen which are easy to compute, usually rectangular patterns or other patterns where t is basically a rounding-operator. Each bucket B_j contains a data structure which contains the points \mathbf{p}_i with $t(\mathbf{p}_i) = j$. Filling all n points $\mathbf{p}_i \in P$ into the bucket array B takes an effort linear in n .

Figure D.4 shows an example of the NNS search with a bucket algorithm on a rectangular grid. A query for the nearest neighbor of a point \mathbf{q} is performed by identifying the bucket B_j which contains \mathbf{q} and finding the nearest neighbor among the points which are in that bucket (step 1). If the bucket is empty, the closest bucket B_k is searched which is not empty and the nearest neighbor is identified among B_k 's points (steps 2–4). When a closest point \mathbf{p}_j is found, either in B_j or B_k , all other buckets which potentially contains points closer than \mathbf{p}_j and have not been already visited need to be tested for closer points (step 5). It can be shown, that for equally distributed points \mathbf{p}_i and n buckets, at average $O(1)$ buckets are visited when searching the nearest neighbor of any point \mathbf{q} [Bentley *et al.*, 1980]. In situations where more than one point lies

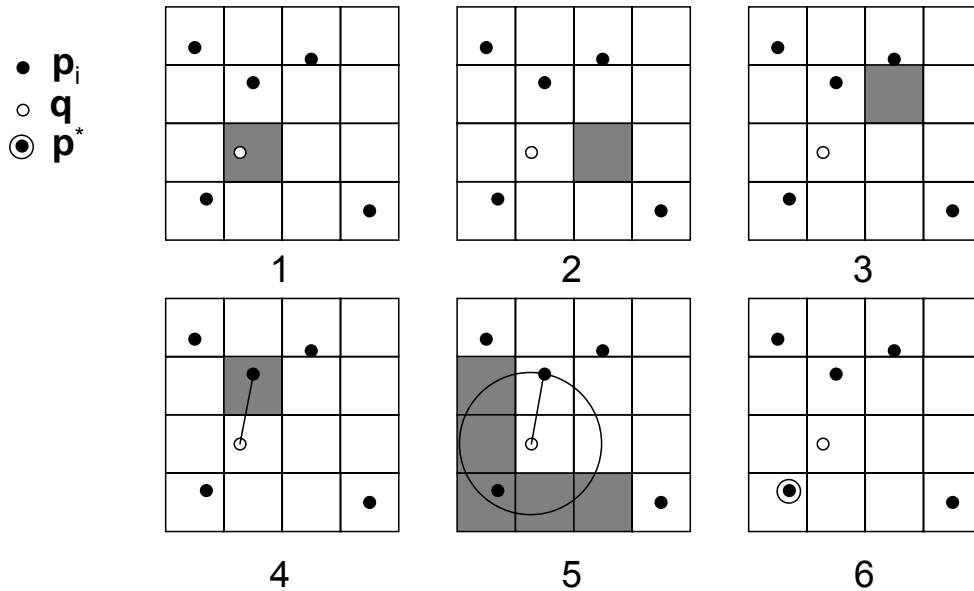


Figure D.4: Example for NNS with bucket search: The query point \mathbf{q} is sorted into the two-dimensional bucket array (1) and the closest not-empty bucket is identified (2–4) and the nearest neighbor \mathbf{p} to \mathbf{q} is identified in the bucket. All buckets which lie under the circle around \mathbf{q} through \mathbf{p} are tested for closer neighbors (5). \mathbf{p}^* is identified as the closest neighbor to \mathbf{q} (6).

in one bucket, the NNS problem has to be solved for the points in one bucket when evaluating the distance function for a bucket. An optimal algorithm, i.e. an algorithm with a computational complexity of $O(n \log n)$ has to be applied within the buckets in order to limit the worst-case behavior of bucket search to $O(n \log n)$.

For the detection of loops, the NNS problem is adjusted as follows:

In a infinite sequence of points $P = \{\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \dots\}$, identify for every point \mathbf{p}_i the first point \mathbf{p}_{i-j} , $j \in [a, n]$ which is closer to \mathbf{p}_i under the Euclidean metric $d(\mathbf{p}_1, \mathbf{p}_2)$ than a threshold $\delta \in \mathbb{R}$ with $a \in [1, n]$ being the smallest integer for which $d(\mathbf{p}_i, \mathbf{p}_{i-a}) > c$ with a cutoff constant $c \in \mathbb{R}$.

The search is performed with a bucket search algorithm with the following adaptations:

- Each bucket is a cuboid with constant dimensions c_x, c_y, c_z .
- Inside each bucket b , only one position \mathbf{p}_b is stored. When multiple input points are sampled in the same bucket b , the mean of their positions is stored.
- A bidirectionally linked list contains all buckets which contain a position in the order in which they were visited. The elements of that list are called e , the bucket one element e refers to is called $e.b$. The oldest element of that list is called the tail of the list, the newest element is called the its head.
- Each element e of that list contains a flag `leftBehind` which is set when the position $\mathbf{p}_{e.b}$ stored in $b.e$ has or has had a distance larger than c from the head. In Figure D.5, this property is represented in the color of the nodes: Nodes which are close to the *head* of the sequence are white, those points which have been left behind are black.
- Points which are added to the buffer are continuously added to both data structures, the bucket array and the bidirectional list. All new points are labeled `leftBehind = false`. Whenever a new point is added, the `leftBehind` flag is updated for the points where it needs to be updated.
- Points which are removed from the buffer are also removed from the bucket array and the bidirectional list.
- When a point \mathbf{p}_i is added, its closest neighbor \mathbf{p}_j within the radius δ is identified which is labeled `leftBehind = true`. When such a point is identified, the sequence of points between \mathbf{p}_j and \mathbf{p}_i is forwarded to the result message thread for output as a detected loop gesture. All elements but \mathbf{p}_i are then removed from the bucket array and the list.

The "condensation" of all points which fall into the same bucket to their mean position results in a subsampling of the input points. It was introduced to rule out the search effort added by the identification of the nearest neighbor inside one bucket. Potentially, this results in two problems:

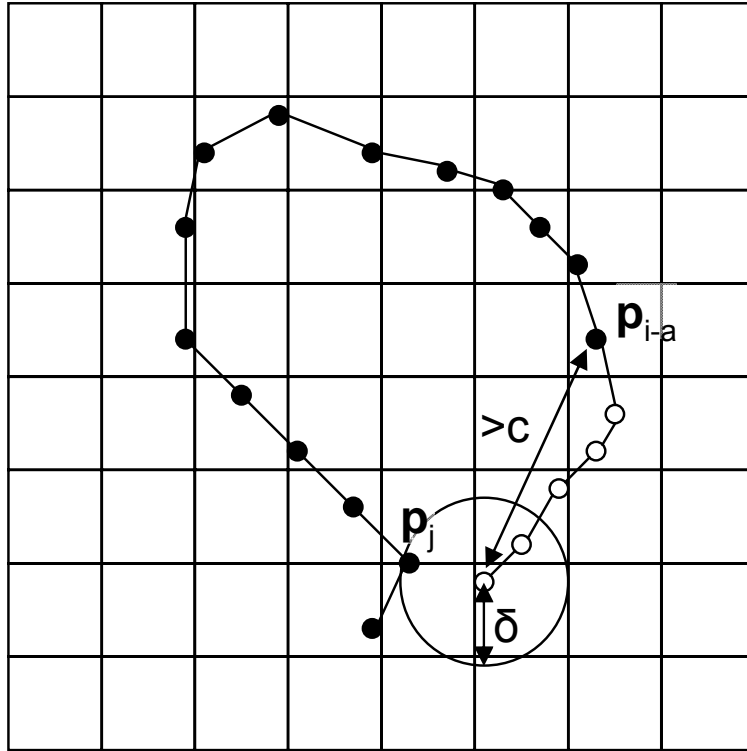


Figure D.5: Bucket search on the nodes of a bidirectional list to identify loops.

- **Misclustering.** That happens, when the path of the tracking coordinates intersects one bucket twice. Instead of adding a new item to the bidirectional list, the position of the existing item is altered. The topology of the list does then no longer match with the real topology of the path it approximates.
- **Aliasing:** Due to the low-pass property of the positions in the buckets, the intersection test may miss intersections or report "near misses" as intersections.

Both effects can easily be compensated or reduced to a negligible level by choosing a bucket size well below the intersection threshold c .

Detection Job Control

The Gesture Detection Module can monitor several tracking streams at once. For each tracked object, a detection job is initialized which contains its own ring buffer, stream receiver, gesture detection threads, and response message sender. The central controller object of the software is the detection job control which generates, maintains and destroys the detection jobs. This module acts a remote method server in the TiCoLi network. The following methods can be called by remote applications:

- **AddDetectionJob** A new detection job is initialized. The calling application has to specify the tracking module and stream name as well as the numeric

parameters of the gesture detection algorithms. The calling device will automatically be added to the detection job's receivers list, i.e. it will receive status messages whenever a gesture has been detected.

- `AddReceiver` A remote application adds itself to the receivers list of an existing detection job.
- `RemoveReceiver` a remote application removes itself from the list of receivers of an existing detection job. The job will be closed when the last application withdraws.
- `SendJobList` a list of the currently running Detection Jobs is sent to the caller of this method.

The detection job instances which are initialized by the detection job control creates, initializes and maintains its receiving, monitoring and sending threads. When one of the threads encounters problems, an event-handling method in the detection job will be called. Possible events are connectivity problems in the stream receiver or gesture detection threads which were overrun by the incoming tracking information, i.e. the reading-iterator of one gesture detection thread was lapped by the writing-iterator of the stream receiver. The detection job and detection job control initialize the appropriate reactions to such events (notification of receivers, adaptation of frame rate, attempt to re-connect, termination of job, ...).

Appendix E

S-DICOM IODs

E.1 The Surface Segmentation IOD

IE	Module	Usage
Patient	Patient	Mandatory
	Specimen	Optional
	Clinical Trial Subject	Optional
Study	General Study	Mandatory
	Patient Study	Optional
	Clinical Trial Study	Optional
Series	General Series	Mandatory
	Segmentation Series	Mandatory
	Clinical Trial Series	Optional
Frame of Reference	Frame of Reference	Mandatory
Equipment	General Equipment	Mandatory
	Enhanced General Equipment	Mandatory
Series	Surface Segmentation	Mandatory
	Surface Mesh	Mandatory
	Common Instance Reference	Required if the surface has been derived from another SOP Instance
	SOP Common	Mandatory

Table E.1: Module table for the surface segmentation IOD.

Attribute Name	Type	Attribute Description
> Include 'Content Identification Macro'		
Segment Sequence	1	Describes the segments that are contained within the data. One or more Items shall be present.
> Segment Number	1	Identification number of the segment. Uniquely identifies a segment within the SOP Instance.
> Segment Label	1	User-defined label identifying this segment. This may be the same as the Code Meaning of the Segmented Property Type Code Sequence.
> Segment Description	3	User-defined description for this segment.
> Segment Algorithm Type	1	Type of algorithm used to generate the segment. Enumerated Values are: AUTOMATIC = calculated segment SEMIAUTOMATIC = calculated segment with user assistance MANUAL = user-entered segment
> Include 'General Anatomy Macro'		
> Segmented Property Category Code Sequence	1	Sequence defining the general category of this segment. This sequence shall contain one item.
>> Include 'Code Sequence Macro'		
> Segmented Property Type Code Sequence	1	Sequence defining the specific property type of this segment. This sequence shall contain one item.
>> Include 'Code Sequence Macro'		
> Surface Count	1	The number of surfaces which comprise this segment. Shall be greater than zero.
> Referenced Surface Sequence	1	Sequence referencing the surfaces composed to construct this segment. The number of Items shall equal the value of Surface Count.
>> Referenced Surface Number	1	Identifies the Surface Number within the Surface Sequence to which this reference applies.
>> Segment Surface Generation Algorithm Identification Sequence	1	A description of how this segment surface was derived.
>>> Include 'Algorithm Identification Macro'		

Table E.2: Surface segmentation module attributes. (Continued on next page)

Attribute Name	Type	Attribute Description
>> Segment Surface Source Instance Sequence	2	A Sequence that identifies the set of Instances by their SOP Class/Instance pair that were used to derive this segment surface. Zero or more items shall be included in this Sequence.
>>> <i>Include 'SOP Instance Reference Macro'</i>		

Table E.2: Surface segmentation module attributes.

Attribute Name	Type	Attribute Description
Number of Surfaces	1	Number of surfaces contained in the Instance. Shall be 1 or more. Shall be the same as the number of Items in Surface Sequence.
Surface Sequence	1	The surfaces that are described within the data. There shall be Number of Surfaces Items in the sequence.
> Surface Number	1	Identification number of the surface. Uniquely identifies a surface within this SOP instance. Shall start at a value of 1, and increase monotonically by 1.
> Surface Comments	3	User-defined comments describing the surface.
> Surface Processing	2	Specifies whether the surface has been modified subsequent to the original generation of the surface. Enumerated Values: YES NO
> Surface Processing Ratio	2C	The Ratio of Remaining points to Original points after processing. Required if Surface Processing is YES.
> Surface Processing Description	3	A textual description of the surface processing performed.
> Surface Processing Algorithm Identification Sequence	2C	Describes the processing method. Required if Surface Processing is YES.
>> <i>Include 'Algorithm Identification Macro'</i>		

Table E.3: Surface mesh module attributes. (Continued on next page)

Attribute Name	Type	Attribute Description
> Recommended Display Grayscale Value	1	A default single gray unsigned value in which it is recommended that the maximum pixel value in this surface be rendered on a monochrome display. The units are specified in P-Values from a minimum of 0000H (black) up to a maximum of FFFFH (white). Note: The maximum P-Value for this Attribute may be different from the maximum P-Value from the output of the Presentation LUT, which may be less than 16 bits in depth.
> Recommended Display CIELab Value	1	A default triplet value in which it is recommended that the surface be rendered on a color display. The units are specified in PCS-Values, and the value is encoded as CIELab.
> Recommended Presentation Opacity	1	Specifies the opacity in which it is recommended that the surface be rendered.
> Recommended Presentation Type	1	Specifies the presentation type in which it is recommended that the surface be rendered.
> Finite Volume	1	Indicates, whether the surface represents a solid ("waterproof") object with an outside and an inside. Enumerated Values: YES = Contains a finite volume NO = Does not contain a finite volume UNKNOWN = Might or might not contain a finite volume
> Manifold	1	Indicates whether the surface is describing an n-1 dimensional manifold in the underlying n-dimensional vector space. Enumerated Values: YES = Manifold in every point NO = Does contain non-manifold points UNKNOWN = Might or might not contain non-manifold points
> Surface Points Sequence	1	The point positions representing vertices of the surface. Only one item shall be permitted in the sequence.
>> <i>Include 'Points Macro'</i>		

Table E.3: Surface mesh module attributes. (Continued on next page)

Attribute Name	Type	Attribute Description
> Surface Points Normals Sequence	2	The normals on the surface for each point. Only one item shall be permitted in the sequence.
>> <i>Include 'Vectors Macro'</i>		The Number of Vectors shall equal Number of Points in this Surface Sequence Item .The Vector Dimensionality shall be 3. If Finite Volume is YES, the normals of the vertices shall point toward the outside of the object. If Finite Volume is not YES, the direction of the normals shall be consistent where possible.
> Surface Mesh Primitives Sequence	1	Only one item shall be permitted in the sequence.
>> <i>Include 'Surface Mesh Primitives Macro'</i>		The primitives' indices shall not exceed Number of Points in this Surface Sequence Item.

Table E.3: Surface mesh module attributes.

Attribute Name	Type	Attribute Description
Algorithm Family Code Sequence	1	The family of algorithm(s) that best describes the software algorithm used. Only one item shall be permitted in the sequence.
> <i>Include 'Code Sequence Macro'</i>		
Algorithm Name Code Sequence	3	The code assigned by a manufacturer to a specific software algorithm. Only one item shall be permitted in the sequence.
>> <i>Include 'Code Sequence Macro'</i>		
Algorithm Name	1	The name assigned by a manufacturer to a specific software algorithm.
Algorithm Version	1	The software version identifier assigned by a manufacturer to a specific software algorithm.
Algorithm Parameters	3	The input parameters used by a manufacturer to configure the behavior of a specific software algorithm.

Table E.4: Algorithm code macro.

Attribute Name	Type	Attribute Description
Number Of Points	1	Specifies the number of points in the point set.
Point Coordinates Data	1	Byte stream containing all points as single precision floats in an x-y-z order.
Point Position Accuracy	3	A single standard deviation of the error for all the points' spatial positions. The units shall be the same as the units of the coordinate system in which the point coordinates are specified.
Average Point Distance	3	The average point distance of the point set. It is given by the average of the distances to the nearest neighbor over all points. The units shall be the same as the units of the coordinate system in which the point coordinates are specified.
Maximum Point Distance	3	The maximum distance of one point to its nearest neighbor. The units shall be the same as the units of the coordinate system in which the point coordinates are specified.
Points Bounding Box Coordinates	3	Two 3D locations defining the cuboid bounding box, parallel to the coordinate system axes, encompassing the point set.
Axis of Rotation	3	A 3D location that combined with Center of Rotation specifies the preferred axis of rotation of this object.
Center of Rotation	1C	A 3D location defining the preferred center of rotation for this point set. Required if Axis of Rotation is present. May be present otherwise.

Table E.5: Points macro.

Attribute Name	Type	Attribute Description
Number of Vectors	1	The number of vectors in the Vector Coordinate Data.
Vector Dimensionality	1	The dimensionality of the underlying vector space.

Table E.6: Vectors macro. (Continued on next page)

Attribute Name	Type	Attribute Description
Vector Accuracy	3	A single standard deviation for all the vectors' coordinates. The units shall be the same as the units of the coordinate system in which the vector coordinates are specified
Vector Coordinate Data	1	A data stream of coordinates encoded as single precision floats in an x-y-z order.

Table E.6: Vectors macro.

Attribute Name	Type	Attribute Description
Vertex Point Index List	2	Byte stream containing a list of point indices where each index defines one vertex primitive.
Edge Point Index List	2	Byte stream containing a list of point indices where each pair of consecutive indices define one line primitive.
Triangle Point Index List	2	Byte stream containing a list of point indices where each triple of consecutive indices define one triangle primitive.
Triangle Strip Sequence	2	All Triangle Strips in this Surface. Zero or more Items shall be present.
> Primitive Point Index List	1	Byte stream containing a list of point indices which define one primitive.
Triangle Fan Sequence	2	All Triangle Fans in this Surface. Zero or more Items shall be present.
> Primitive Point Index List	1	Byte stream containing a list of point indices which define one primitive.
Line Sequence	2	All Lines in this Surface. Zero or more Items shall be present.
> Primitive Point Index List	1	Byte stream containing a list of point indices which define one primitive.
Facet Sequence	2	All Facets in this Surface. Each sequence Item describes one facet. Zero or more Items shall be present.
> Primitive Point Index List	1	Byte stream containing a list of point indices which define one primitive.

Table E.7: Surface mesh primitives macro.

E.2 The Implant Template IOD

IE	Module	Usage
Implant Template	Generic Implant Template Description	Mandatory
	Generic Implant Template Derivation and Versioning	Mandatory
	Generic Implant Template 2D Drawings	Required if Generic Implant Template 3D Models Module is not present
	Generic Implant Template 3D Models	Required if Generic Implant Template 2D Drawings Module is not present.
	Generic Implant Template Mating Features	Optional
	Generic Implant Template Planning Landmarks	Optional
	SOP Common	Mandatory
Surface Mesh	Surface Mesh	Required if Generic Implant Template 3D Models Module is present.

Table E.8: Module table for the generic implant template IOD.

Attribute Name	Type	Attribute Description
Manufacturer	1	Name of the manufacturer that produces the implant.
Implant Name	1	The (product) name of the implant.
Implant Size	1C	The size descriptor of the component. Shall be present if the component exists in different sizes and the size number is not part of the name or identifier. May be present otherwise.
Implant Part Number	1	The (product) identifier of the implant.
Implant Target Anatomy Sequence	3	Sequence that identifies the anatomical region the implant is to be implanted to. One or more Items shall be present.
> <i>Include 'Code Sequence Macro'</i>		

Table E.9: Generic implant template description module attributes. (Continued on next page)

Attribute Name	Type	Attribute Description
Notification from Manufacturer Sequence	1C	Required if the manufacturer has issued a critical notification, recall, or has obsoleted the implant or implant template. One or more Items shall be present.
> Information Issue Date	1	Date information was issued.
> Information Summary	1	Summary of the information.
> MIME Type of Encapsulated Document	2	The type of the encapsulated document stream described using the MIME Media Type (see RFC 2046). Mime Type shall be "application/pdf"
> Encapsulated Document	2	Encapsulated Document stream, containing a document encoded according to the MIME Type. The complete manufacturer notification describing the template.
Information From Manufacturer Sequence	3	One or more Items shall be present in the sequence.
> Information Issue Date	1	Date information was issued.
> Information Summary	1	Summary of the information.
> MIME Type of Encapsulated Document	2	The type of the encapsulated document stream described using the MIME Media Type (see RFC 2046). Mime Type shall be "application/pdf"
> Encapsulated Document	3	Encapsulated Document stream, containing a document encoded according to the MIME Type. The complete manufacturer information.
Implant Regulatory Disapproval Code Sequence	1C	Sequence containing countries and regions in which the implant is not approved for usage. Required if the implant has been disapproved in a country or a region. If present, one or more Items shall be present in the sequence.
> <i>Include 'Code Sequence Macro'</i>		
Overall Template Spatial Tolerance	2	Tolerance applying to all measurements, locations and dimensions in this Implant Template

Table E.9: Generic implant template description module attributes. (Continued on next page)

Attribute Name	Type	Attribute Description
Materials Code Sequence	1	A code sequence specifying the materials the implant was built from. One or more Items shall be present in the Sequence.
> Include 'Code Sequence Macro'		
Coating Materials Code Sequence	1C	Required if the implant is coated. A code sequence specifying the materials the implant is coated with. One or more Items shall be present in the Sequence.
> Include 'Code Sequence Macro'		
Implant Types Code Sequence	1	Sequence containing a coded description of the type of implant the template reflects. One or more Items shall be present in the Sequence.
> Include 'Code Sequence Macro'		
Fixation Method Sequence	1	The method which will be used to fixate the implant in the body. Only a single Item shall be permitted in this sequence.
> Include 'Code Sequence Macro'		

Table E.9: Generic implant template description module attributes.

Attribute Name	Type	Attribute Description
Implant Template Version	1	The version code of the implant template. If Implant Type is DERIVED, this shall have the same value as the Implant Template Version of the manufacturer's implant template from which this instance was derived.
Effective Date	1	Date and time from which this Instance is or will be valid.
Replaced Implant Template	1C	Reference to the Implant Template which is replaced by this template. Shall be present if this Instance replaces another Instance. Only one Item may be present.
> Include 'SOP Instance Reference Macro'		
Implant Type	1	Indicates whether the Implant Template is derived from another Implant Template. Enumerated Values: ORIGINAL DERIVED

Table E.10: Generic implant template derivation and versioning module attributes. (Continued on next page)

Attribute Name	Type	Attribute Description
Original Implant Template	1C	Reference to the Implant Template Instance with Implant Type ORIGINAL from which this Instance was ultimately derived. Shall be present if Implant Type is DERIVED. Only one Item may be present.
> Include 'SOP Instance Reference Macro'		
Derivation Implant Template	1C	Reference to Implant Template Instance from which this Instance was directly derived. Shall be present if Implant Type is DERIVED. Only one Item may be present.
> Include 'SOP Instance Reference Macro'		

Table E.10: Generic implant template derivation and versioning module attributes.

Attribute Name	Type	Attribute Description
HPGL Document Sequence	1	The 2D template representations of this implant. If present, one or more Items shall be present in the sequence.
> 2D Drawing ID	1	Identification number of the HPGL Document. Uniquely identifies a HPGL Document within this SOP instance. Shall start at a value of 1, and increase monotonically by 1.
> HPGL Document Label	3	Label assigned to that document.
> View Orientation Code Sequence	1	Coded description of the direction of view represented by this 2D template. This sequence shall contain one item.
>> Include 'Code Sequence Macro'		
> View Orientation Modifier	3	Direction Cosines of the view direction represented by the 2D template relative to the base direction defined in the View Orientation Code Sequence
> HPGL Document Scaling	1	Conversion factor [real world mm/printed mm] See
> HPGL Document	1	The HPGL document as a plain byte stream. See
> HPGL Contour Pen Number	1	Number of pen used in the encapsulated HPGL document for outlines.

Table E.11: Generic implant template 2D drawings module attributes. (Continued on next page)

Attribute Name	Type	Attribute Description
> HPGL Pen Sequence	1	Sequence containing labels for each pen used in the encapsulated HPGL Document. Shall contain one item per pen used in the HPGL document.
>> HPGL Pen Number	1	Number of the pen in the HPGL document
>> HPGL Pen Label	1	Label of that pen.
>> HPGL Pen Description	3	Description of the kind of information drawn with this pen.
> Recommended Rotation Point	1	Point around which the 2D template is rotated in manual planning.
> Bounding Rectangle	1	Coordinates of the smallest rectangle parallel to the paper axes that contains the whole drawing.

Table E.11: Generic implant template 2D drawings module attributes.

Attribute Name	Type	Attribute Description
Frame of Reference UID	1	Identifies a Frame of Reference for this component.
Contour Surface Number Reference	1C	Surface Number of the surface that represents the shape of the implant. Shall only be present if Number of Surfaces is present.
Surface Model Parameter Sequence	1C	Required if Number Of Surfaces is present. Shall contain one Item per Item in the Surface Sequence
> Referenced Surface Number	1	Reference to a Surface Number present in the Surface Sequence
> Surface Model Label	1	Label for this surface.
Surface Model Scaling Factor	1C	Scaling factor [mm/Surface unit] Shall only be present if Number of Surfaces is present.

Table E.12: Generic implant template 3D models module attributes.

Attribute Name	Type	Attribute Description
Mating Feature Sets Sequence	3	Defines a number of feature sets used to combine the implant with other implants. If present, one or more Items shall be present in the Sequence.

Table E.13: Generic implant template mating features module attributes. (Continued on next page)

Attribute Name	Type	Attribute Description
> Mating Feature Set ID	1	Identification number of the set. Uniquely identifies a mating feature set within this SOP instance. Shall start at a value of 1, and increase monotonically by 1.
> Mating Feature Set Label	1	Label of the feature set.
> Mating Feature Sequence	1	The mating features of the set. One or more Items shall be present in the Sequence.
>> Mating Feature ID	1	Identification number of the mating feature. Uniquely identifies a mating feature within this Sequence Item.
>> 3D Mating Point	1C	Origin of the contact system Required if Mating Feature 2D Coordinates Sequence is not present and Contour Surface Number Reference is present. May be present if Mating Feature 2D Coordinates Sequence is present and Contour Surface Number Reference is present.
>> 3D Mating Axes	1C	Direction cosines of the contact system Required if 3D Mating Point is present.
>> Mating Feature 2D Coordinates Sequence	1C	Sequence containing the coordinates of the mating feature in the HPGL documents. Required if 3D Mating Point is not present and HPGL Document Sequence is present. May be present if 3D Mating Point is present and HPGL Document Sequence is present. If present, one or more Items shall be present in the sequence.
>>> Referenced 2D Drawing ID	1	Reference to a 2D Drawing ID present in the HPGL Document Sequence. Shall be unique within the sequence.
>>> 2D Mating Point	1	Origin of the contact system
>>> 2D Mating Axes	1	Direction cosines of the contact system
>> Mating Feature Degree of Freedom Sequence	3	Sequence containing the degrees of freedom in this mating feature. If present, one or more Items shall be present in the Sequence.

Table E.13: Generic implant template mating features module attributes. (Continued on next page)

Attribute Name	Type	Attribute Description
>>> Degree of Freedom ID	1	Identification number of the degree of freedom. Uniquely identifies a degree of freedom within this Sequence Item. Shall start at a value of 1, and increase monotonically by 1.
>>> Degree of Freedom Type	1	Indicates the type of the degree of freedom. Enumerated Values TRANSLATION ROTATION
>>> Degree of Freedom 3D Axis	1C	Direction cosines of the axis of the degree of freedom in the FOR of the template. See C.X.1.1.4 Required if 3D Mating Point is present.
>>> 3D Range of Freedom	1C	2 floats defining an interval for this degree of freedom. See C.X.1.1.4 Required if 3D Mating Point is present.
>>> 2D Degree of Freedom Sequence	1C	Sequence containing the geometric specifications of the degrees of freedom for this HPGL Document. Required if Mating Feature 2D Coordinates Sequence is present. If present, one or more Items shall be present in the sequence.
>>>> Referenced 2D Drawing ID	1	Reference to a 2D Drawing ID present in the HPGL Document Sequence. Shall be unique within the sequence.
>>>> Degree Of Freedom 2D Axis	1	Direction cosines of the axis of the degree of freedom.
>>>> 2DRange of Freedom	1	Interval of freedom for this degree of freedom.

Table E.13: Generic implant template mating features module attributes.

Attribute Name	Type	Attribute Description
Planning Landmark Point Sequence	3	Sequence containing point landmarks. If present, one or more Items shall be present in the sequence.
> Include 'Planning Landmark Point Macro'		

Table E.14: Generic implant template planning landmarks module attributes.
(Continued on next page)

Attribute Name	Type	Attribute Description
> Planning Landmark ID	1	Identification number of the planning landmark. Uniquely identifies a planning landmark within this SOP instance. Shall start at a value of 1, and increase monotonically by 1.
> Planning Landmark Description	3	Description of the purpose or intended use of this landmark.
> Planning Landmark Identification Sequence	2	Coded Description of the real-world point which is represented by this landmark. If present, one or more items shall be present in the sequence.
>> <i>Include 'Code Sequence Macro'</i>		
Planning Landmark Line Sequence	3	Sequence containing line landmarks. If present, one or more Items shall be present in the sequence.
> <i>Include 'Planning Landmark Line Macro'</i>		
> Planning Landmark ID	1	Identification number of the planning landmark. Uniquely identifies a planning landmark within this SOP instance. Shall start at a value of 1, and increase monotonically by 1.
> Planning Landmark Description	3	Description of the purpose or intended use of this landmark.
> Planning Landmark Identification Sequence	2	Coded Description of the real-world line which is represented by this landmark. Shall contain one or more Items if present.
>> <i>Include 'Code Sequence Macro'</i>		
Planning Landmark Plane Sequence	3	Sequence containing plane landmarks. If present, one or more Items shall be present in the sequence.
> <i>Include 'Planning Landmark Plane Macro'</i>		
> Planning Landmark ID	1	Identification number of the planning landmark. Uniquely identifies a planning landmark within this SOP instance. Shall start at a value of 1, and increase monotonically by 1.
> Planning Landmark Description	3	Description of the purpose or intended use of this landmark.

Table E.14: Generic implant template planning landmarks module attributes.
(Continued on next page)

Attribute Name	Type	Attribute Description
> Planning Landmark Identification Sequence	2	Coded Description of the real-world plane which is represented by this landmark. Shall contain one or more Items if present.
>> <i>Include 'Code Sequence Macro'</i>		

Table E.14: Generic implant template planning landmarks module attributes.

Attribute Name	Type	Attribute Description
2D Point Coordinates Sequence	1C	Sequence containing the 2D coordinates of the point in the HPGL documents. Required if 3D Point Coordinates is not present and HPGL Document Sequence is present. May be present if 3D Point Coordinates is present and HPGL Document Sequence is present. If present, one or more Items shall be present in the Sequence.
> Referenced 2D Drawing ID	1	Reference to a 2D Drawing ID present in the HPGL Document Sequence. Shall be unique within the sequence.
> 2D Point Coordinates	1	Coordinates of the point in the HPGL document. Coordinates are measured in millimeters of the printing space.
3D Point Coordinates	1C	3D Coordinates of the point. Required if 2D Point Coordinates Sequence is not present and Contour Surface Number Reference is present. May be present if 2D Point Coordinates Sequence is present and Contour Surface Number Reference is present.

Table E.15: Planning landmark point macro.

Attribute Name	Type	Attribute Description
2D Line Coordinates Sequence	1C	Sequence containing the 2D coordinates of the line in the HPGL documents. Required if 3D Line Coordinates is not present and HPGL Document Sequence is present. May be present if 3D Line Coordinates is present and HPGL Document Sequence is present. If present, one or more Items shall be present in the Sequence.

Table E.16: Planning landmark line macro. (Continued on next page)

Attribute Name	Type	Attribute Description
> Referenced 2D Drawing ID	1	Reference to a 2D Drawing ID present in the HPGL Document Sequence. Shall be unique within the sequence.
> 2D Line Coordinates	1	Coordinates of the line in the HPGL document. Coordinates are measured in Millimeters of the printing space.
3D Line Coordinates	1C	3D Coordinates of the line. Required if 2D Line Coordinates Sequence is not present and Contour Surface Number Reference is present. May be present if 2D Line Coordinates Sequence is present and Contour Surface Number Reference is present.

Table E.16: Planning landmark line macro.

Attribute Name	Type	Attribute Description
2D Plane Coordinates Sequence	1C	Sequence containing the 2D coordinates of the plane's intersection with the HPGL documents. Required if 3D Plane Origin is not present and HPGL Document Sequence is present. May be present if 3D Plane Origin is present and HPGL Document Sequence is present. If present, one or more Items shall be present in the Sequence.
> Referenced 2D Drawing ID	1	Reference to a 2D Drawing ID present in the HPGL Document Sequence. Shall be unique within the sequence.
> 2D Plane Intersection	1	2D Coordinates of the intersection of the plane with the projection plane. Coordinates are measured in Millimeters of the printing space.
3D Plane Origin	1C	3D Coordinates of the plane origin. Required if 2D Plane Coordinates Sequence is not present and Contour Surface Number Reference is present. May be present if 2D Plane Coordinates Sequence is present and Contour Surface Number Reference is present.
3D Plane Normal	1C	3D Coordinates of the plane normal. Required if 3D Plane Origin is present.

Table E.17: Planning landmark plane macro.

E.3 The Implant Assembly Template IOD

IE	Module	Usage
Implant Assembly Template	Implant Assembly Template	Mandatory
	SOP Common	Mandatory

Table E.18: Module table for the implant assembly template IOD.

Attribute Name	Type	Attribute Description
Implant Assembly Template Name	2	A name given to the assembly described in this instance.
Implant Assembly Template Issuer	1	The person or organization who issued the assembly template.
Effective Date	1	Date from which on this Instance is valid
Implant Assembly Template Version	2	The version code of the Implant Assembly Template.
Replaced Implant Assembly Template	1C	Reference to the Implant Assembly Template which is replaced by this Instance. Shall be present if this Instance replaces another Instance. If present, exactly one Item shall be present in the Sequence
> Include 'SOP Instance Reference Macro'		
Implant Assembly Template Type	1	Indicates whether the Implant Assembly Template is derived from another Instance. Enumerated Values: ORIGINAL DERIVED
Original Implant Assembly Template	1C	Reference to the Implant Assembly Template Instance with Implant Assembly Template Type ORIGINAL from which this Instance was ultimately derived. Shall be present if Implant Assembly Template Type is DERIVED. If present, exactly one Item shall be present in the Sequence
> Include 'SOP Instance Reference Macro'		

Table E.19: Implant assembly template module attributes. (Continued on next page)

Attribute Name	Type	Attribute Description
Derivation Implant Assembly Template	1C	Reference to the Implant Template Instance from which this Instance was directly derived. Shall be present if Implant Assembly Template Type is DERIVED. If present, exactly one Item shall be present in the Sequence
> Include 'SOP Instance Reference Macro'		
Implant Assembly Template Target Anatomy Sequence	1	Sequence that identifies the anatomical region the implant assembly is to be implanted to. One or more Items shall be present in the Sequence.
> Include 'Code Sequence Macro'		
Procedure Type Code Sequence	1	Coded description of the procedure by which the assembly is implanted. One or more Items shall be present in the Sequence.
> Include 'Code Sequence Macro'		
Surgical Technique	3	Name of the surgical technique associated with this assembly template.
> MIME Type of Encapsulated Document	2	The type of the encapsulated document stream described using the MIME Media Type (see RFC 2046). Mime Type shall be "application/pdf"
> Encapsulated Document	2	Encapsulated Document stream, containing a document encoded according to the MIME Type. PDF description of the surgical technique.
Component Types Sequence	1	Sequence containing sets of component of which the assembly is constructed. One or more Items shall be present in the Sequence.
> Component Type Label	1	Label assigned to that type of component.
> Exclusive Component Type	1	When YES only one sequence item of Component Sequence may be used in a valid assembly. Defined Terms: YES NO

Table E.19: Implant assembly template module attributes. (Continued on next page)

Attribute Name	Type	Attribute Description
> Mandatory Component Type	1	When YES, at least one sequence item of Component Sequence must be used in a valid assembly. Defined Term: YES NO
> Component Sequence	1	Sequence containing references to implant templates used in the assembly. One or more Items shall be present in the Sequence.
> Include 'SOP Instance Reference Macro'		
>> Component ID	1	Assigns an identification number to that Instance for local references. Uniquely identifies a referenced Implant Template Instance within this SOP instance. Shall start at a value of 1, and increase monotonically by 1.
Component Assembly Sequence	3	Sequence containing information about how to connect the implants from the component groups. If present, one or more Items shall be present in the Sequence.
> Component 1 Referenced ID	1	Component ID of the first component of this connection.
> Component 1 Referenced Mating Feature Set ID	1	Identifies the Mating Feature Set ID within the Mating Feature Set Sequence of Component 1 to which this reference applies.
> Component 1 Referenced Mating Feature ID	1	Identifies the Mating Feature ID within the referenced Mating Feature Sequence Component 2 to which this reference applies.
> Component 2 Referenced ID	1	Component ID of the second component of this connection.
> Component 2 Referenced Mating Feature Set ID	1	Identifies the Mating Feature Set ID within the Mating Feature Set Sequence of Component 2 to which this reference applies.
> Component 2 Referenced Mating Feature ID	1	Identifies the Mating Feature ID within the referenced Mating Feature Sequence in Component 2 to which this reference applies.

Table E.19: Implant assembly template module attributes.

E.4 The Implant Template Group IOD

IE	Module	Usage
Implant Template Group	Implant Template Group	Mandatory
	SOP Common	Mandatory

Table E.20: Module table for the implant template group IOD.

Attribute Name	Type	Attribute Description
Implant Template Group Name	1	Name of this group.
Implant Template Group Description	3	Description of purpose or intent of this group.
Implant Template Group Issuer	1	Person or organization which issued this group.
Effective Date	1	Date from which on this Instance is valid.
Implant Template Group Version	2	The version code of the Implant Template Group.
Replaced Implant Template Group	1C	Reference to the Implant Template Group which is replaced by this Instance. Shall be present if this Instance replaces another Instance. Only one Item may be present.
> <i>Include 'SOP Instance Reference Macro'</i>		
Implant Template Group Target Anatomy Sequence	3	Sequence that identifies the anatomical region the implant is to be implanted to. One or more Items shall be present.
> <i>Include 'Code Sequence Macro'</i>		
Implant Template Group Members Sequence	1	Contains references to all Implant Template SOP instances which are part of this group. One or more items shall be present.
> <i>Include 'SOP Instance Reference Macro'</i>		
> Implant Template Group Member ID	1	Assigns an identification number to that Instance for local references. Uniquely identifies a referenced Implant Template Instance within this SOP instance. Shall start at a value of 1, and increase monotonically by 1.

Table E.21: Implant template group module attributes. (Continued on next page)

Attribute Name	Type	Attribute Description
> 3D Implant Template Group Member Matching Point	3	Shall only be present if Number of Surfaces is present in the Instance referenced by Referenced Implant Template UID.
> 3D Implant Template Group Member Matching Axes	1C	Required if 3D Implant Template Group Matching Point is present.
> Implant Template Group Member Matching 2D Coordinates Sequence	3	Shall only be present if HPGL Document Sequence is present in the Instance referenced by Referenced Implant Template UID. If present, one or more Items shall be present in the sequence.
>> Referenced 2D Drawing ID	1	Reference to a 2D Drawing ID present in the HPGL Document Sequence of the Instance which is referenced by Referenced Implant Template UID. Shall be unique within the sequence.
>> 2D Implant Template Group Member Matching Point	1	2D Coordinates of the matching point in the referenced drawing
>> 2D Implant Template Group Member Matching Axes	1	2D Coordinate of the matching axes in the referenced drawing
Implant Template Group Variation Dimension Sequence	1	Sequence that lists all Variation Dimensions that are covered by this group. One or more items shall be present.
> Implant Template Group Variation Dimension Name	1	Descriptive name of the variation dimension.
> Implant Template Group Variation Dimension Rank Sequence	1	Defines the order in which the implant group members are sorted according to this dimension.
>> Referenced Implant Template Group Member ID	1	Identifies one implant group member by reference to one Implant Template Group Member ID. Shall be unique within this Implant Template Group Variation Dimension Rank Sequence.
>> Implant Template Group Variation Dimension Rank	1	Indicates the rank of this Implant Template in this Variation Dimension. In one Implant Template Group Variation Dimension Rank Sequence there may be more than one Implant Templates with the same rank.

Table E.21: Implant template group module attributes.

Bibliography

- 3Mensio. 2009. *3Mensio Valves*. <http://www.3mensio.com/>. [Online; accessed 10-December-2009].
- Abramyuk, A. M., Haase, M. G., & Abolmaali, N. D. 2008. Tumorbildgebung: Morphologisch - Funktionell - Metabolisch - Molekular. *Pages 25–35 of: Niederlag, W., Lemke, H. U., Meixensberger, J., & Baumann, M. (eds), Modellgestützte Therapie*. Health Academy.
- Adams, L., Krybus, W., Meyer-Ebrecht, D., Rueger, R., Gilsbach, J.M., Moesges, R., & Schloendorff, G. 1990. Computer-assisted surgery. *IEEE Computer Graphics and Applications*, **10**(3), 43–51.
- Al Ali, A. M., Altwegg, L., Horlick, E. M., Feindel, C., Thompson, C. R., Cheung, A., Carere, R. G., Humphries, K., Ye, J., Masson, J. B., & Webb, J. G. 2008. Prevention and management of transcatheter balloonexpandable aortic valve malposition. *Catheterization and Cardiovascular Interventions*, **72**(4), 573–578.
- Al-Attar, N., Ghodbane, W., Himbert, D., Rau, C., Raffoul, R., Messika-Zeitoun, D., Brochet, E., Vahanian, A., & Nataf, P. 2009. Unexpected complications of transapical aortic valve implantation. *Annals of Thoracic Surgery*, **88**(1), 90–94.
- Alesch, F. 1994. A Simple Technique for Making a Stereotactic Localiser Both CT and MRI Compatible. *Acta Neurochirurgica*, **127**, 118–120.
- Alon, E., & Schüpfer, G. 1999. Operationssaal-Management. *Anaesthesist*, **48**, 689–697.
- Ambler, S. W. 2004. *The Object Primer : Agile Model-Driven Development with UML 2.0*. 3rd edn. Cambridge University Press.
- Anagnostaki, A., Pavlopoulos, S., & Koutsouris, D. 2001. XML and the VITAL Standard: The Document-oriented Approach for Open Telemedicine Applications. *Pages 77–81 of: Proc. of MEDINFO*.
- Ayache, N., Cinquin, P., Cohen, I., Cohen, L., Leitner, F., & Mongar, O. 1996. Segmentation of Complex Three-Dimensional Medical Objects: A Challenge and a Requirement for Computer-Assisted Surgery Planning and Performance. *Chap. 8, pages 59–76 of: Taylor, R. H., Lavallée, S., Burdea, G. C., & Mösges, R. (eds)*

- Computer Integrated Surgery - Technology and Clinical Applications*. The MIT Press.
- Azari, A., & Nikzad, S. 2008. Computer-assisted implantology: historical background and potential outcomes - a review. *International Journal of Medical Robotics and Computer Assisted Surgery*, **4**(2), 95–104.
- Balachandran, R., Labadie, R. F., & Fitzpatrick, J. M. 2006. Validation of a fiducial frame system for image-guided otologic surgery utilizing BAHA bone screws. *Pages 518–521 of: Proc. of 3rd IEEE International Symposium on Biomedical Imaging: Nano to Macro*.
- Ballanger, B., van Eimeren, T., & Strafella, A. P. 2009. *Diagnostic PET in Image Guided Neurosurgery*. Springer. Chap. 22, pages 308–323.
- Ballantyne, G. H., & Moll, F. 2003. The da Vinci telerobotic surgical system: the virtual operative field and telepresence system. *The Surgical Clinics of North America*, **83**(6), 1293–1304.
- Barnett, G. H., Kormos, D. W., Steiner, C. P., & Weisenberger, J. 1993. Intraoperative localization using an armless, frameless stereotactic wand. *Journal of Neurosurgery*, **78**(3), 510–514.
- Bassingthwaighe, J. B. 2000. Strategies for the physiome project. *Annals of Biomedical Engineering*, **28**(8), 1043–1058.
- Bast, P., Popovic, A., Wu, T., Heger, S., Engelhardt, M., Lauer, W., Radermacher, K., & Schmieder, K. 2006. Robot- and computer-assisted craniotomy: resection planning, implant modelling and robot safety. *The International Journal of Medical Robotics and Computer Assisted Surgery*, **2**(2), 168–178.
- Baumhove, O., & Schröter, K.-H. 2005. Gesundheitsfördernde Aspekte bei der Reorganisation einer zentralen Operationsabteilung. *Gesundheitswesen*, **67**(2), 112–116.
- Belliveau, J. W., Kennedy, Jr., D. N., McKinstry, R. C., Buchbinder, B. R., Weiskoff, R. M., Cohen, M. S., Vevea, J. M., Brady, T. J., & Rosen, B. R. 1991. Functional mapping of the human visual cortex by magnetic resonance imaging. *Science*, **254**(5032), 716–719.
- Bentley, J. L., Weide, B. W., & Yao, A. C. 1980. Optimal Expected-Time Algorithms for Closest Point Problems. *ACM Trans. Math. Softw.*, **6**(4), 563–580.
- Bentley, J. Louis, & Friedman, J. H. 1979. Data Structures for Range Searching. *ACM Comput. Surv.*, **11**(4), 397–409.
- Blecha, S., Lindner, D., Neumuth, T., Trantakis, C., Burgert, O., & Meixensberger, J. 2007. Workflow von intrakraniellen Hirntumoroperationen ohne und mit Neuronavigation. *Pages 259–262 of: CURAC Proceedings*.

- Bockhold, U., Bisler, A., Becker, M., Müller-Wittig, W., & Voss, G. 2003. Augmented Reality for Enhancement of Endoscopic Interventions. *Page 97 of: Proc. of IEEE Virtual Reality Conference.*
- Bohn, S., Lessnau, M., & Burgert, O. 2009 (June). Systems Monitoring and Diagnosis of Digital Operating Room (DOR) Equipment using Supervisory Control and Data Acquisition (SCADA) Technology. *Pages 146–147 of: Lemke, Heinz U., Inamura, Kiyonari, Doi, Kunio, Vannier, Michael W., & Farman, Allan G. (eds), International Journal of Computer Assisted Radiology and Surgery*, vol. 4, supp. 1.
- Bonow, R. O., Carabello, B. A., & Chatterjee, K. 2006. ACC/AHA 2006 Guidelines for the management of patients with valvular heart disease: Executive summary. *Circulation*, **114**, 450–527.
- Brandestini, M.A., Howard, E.A., Eyer, M. and Stevenson, J.G., & Weiler, T. 1979. Visualization of intracardiac defects by M/Q mode echo/Doppler ultrasound. *Circulation*, **60**(2), 12.
- Brownell, G.L., & Sweet, W.H. 1953. Localization of brain tumors with positron emitters. *Nucleonics*, **11**, 40–45.
- Buehler, W. J., Gilfrich, J. V., & Wiley, R. C. 1963. Effect of Low-Temperature Phase Changes on the Mechanical Properties of Alloys near Composition TiNi. *Journal of Applied Physics*, **34**(5), 1475–1477.
- Burgert, O., Neumuth, T., Lempp, F., Mudunuri, R., Meixensberger, J., Strauß, G., Dietz, A., Jannin, P., & Lemke, H.U. 2006 (June). Linking top-level ontologies and surgical workflows. *In: International Journal of Computer Assisted Radiology and Surgery*, vol. 1, supp. 1.
- Burgert, O., Neumuth, T., Gessat, M., Jacobs, S., & Lemke, H. U. 2007. Deriving DICOM surgical extensions from surgical workflows. *In: Horii, S. C., & Andriole, K. P. (eds), SPIE Medical Imaging: PACS and Imaging Informatics*. Presented at the SPIE Conference, vol. 6516, paper 651604.
- Chandler, W. F., Knake, J. E., McGillicuddy, J. E., Lillehei, K. O., & Silver, T. M. 1982. Intraoperative use of real-time ultrasonography in neurosurgery. *Journal of Neurosurgery*, **57**(2).
- Chesler, D. A. 1971. Three-dimensional activity distribution from multiple positron scintigraphs. *Journal of Nuclear Medicine*, **12**, 347–348.
- Clavel, M. A., Webb, J. G., Pibarot, P., Altwegg, L., Dumont, E., Thompson, C., De Larochelière, R., Doyle, D., Masson, J. B., Bergeron, S., Bertrand, O. F., & Rodés-Cabau, J. 2009. Comparison of the hemodynamic performance of percutaneous and surgical bioprostheses for the treatment of severe aortic stenosis. *Journal of the American College of Cardiology*, **53**(20), 1883–1891.

- Cleary, K. R. 1999. *Technical Requirements for Image-Guided Spine Procedures - Workshop Report*. Tech. rept. ISIS Center, Department of Radiology, Georgetown University Medical Center.
- Cleary, K. R., & Kinsella, A. 2004. *OR2020 The Operating Room of the Future - Workshop Report*. Tech. rept. ISIS Center, Department of Radiology, Georgetown University Medical Center.
- Cosman, Eric R., & Roberts, Theodore S. 2002. *CT and MRI visible index markers for stereotactic localization*. Patent US. 6,419,680 B1.
- dcm4chee. 2005. *Open Source Clinical Image and Object Management*. <http://www.dcm4chee.org>. [Online; accessed 26-October-2009].
- Deinhardt, M. 2003. Manipulators and integrated OR systems - requirements and solutions. *Minimally Invasive Therapy and Allied Technologies*, **12**(6), 284 – 292.
- Di Gioia, A. M., Kanade, T., & Wells, P. 1996. Final Report For The Second International Workshop On Robotics And Computer Assisted Medical Interventions. *Pages 69–101 of: Computer Aided Surgery*, vol. 2.
- DICOM@OFFICE. 2005. *DCMTK - DICOM Toolkit*. <http://dicom.offis.de/dcmtk.php.de>. [Online; accessed 27-April-2009].
- Dijkstra, E. W. 1982. EWD 447: On the role of scientific thought. *Selected Writings on Computing: A Personal Perspective*, 60–66.
- Dreyer, K. J., Hirschorn, D. S., Thrall, J. H., & Mehta, A. 2005. *PACS: A Guide to the Digital Revolution*. 2 edn. Springer. ISBN 0387260102.
- Dützmann, S. 2009. *BASICS Neurochirurgie*. 1. edn. München, Jena: Elsevier Urban & Fisher. ISBN 978-3-437-42486-1.
- Eisner, W. 2001. Elektrophysiologisches Neuromonitoring in der Neurochirurgie. *Journal für Neurologie, Neurochirurgie und Psychiatrie*, **2**(3), 38–55.
- ENV. 2000. *ENV 13734 - Health informatics – Vital signs information representation*.
- Faddis, M. N., Blume, W., Finney, J., Hall, A., Rauch, J., Sell, J., Bae, K. T., Talcott, M., & Lindsay, B. 2002. Novel, Magnetically Guided Catheter for Endocardial Mapping and Radiofrequency Catheter Ablation. *Circulation*, **106**(12), 2980–2985.
- Falk, V., Gummert, J. F., Walther, T., Hayase, M., Berry, G. J., & Mohr, F. W. 1999. Quality of computer enhanced totally endoscopic coronary bypass graft anastomosis - comparison to conventional technique. *European Journal of Cardio-Thoracic Surgery*, **15**(3), 260–265.
- Falk, V., Diegeler, A., Walther, T., Banusch, J., Brucerius, J., Raumans, J., Autschbach, R., & Mohr, F. W. 2000. Total endoscopic computer enhanced coronary artery bypass grafting. *European Journal of Cardio-Thoracic Surgery*, **17**(1), 38–45.

- Fasano, V. A., Urciuoli, R., Ponzio, R. M., & Lanotte, M. M. 1986. The effects of new technologies on the surgical management of brainstem tumors. *Surgical Neurology*, **25**(3), 219–226.
- Fitzgerald, G., & Swanson, J. 1992. Measuring the effects of laboratory automation: The power of empirically derived models. *The Journal of Automatic Chemistry*, **14**(2), 55–57.
- Friets, E.M., Strohbehn, J.W., Hatch, J.F., & Roberts, D.W. 1989. A frameless stereotaxic operating microscope for neurosurgery. *IEEE Transactions on Biomedical Engineering*, **36**(6), 608–617.
- Galloway, R., & Peters, T. 2008. *Overview and History of Image-Guided Interventions*. Springer US. Chap. 1, pages 1–21.
- Gane, C., & Sarson, T. 1979. *Structured Systems Analysis : Tools and Techniques*. Englewoodd Cliff, New Jersey: Prentice-Hall, Inc.
- Garfagni, H., & Klipfel, B. 1995. Integrating HIS and PACS: The DICOM Solution. *Pages 438–444 of: Proceedings on the International Symposium on Computer Assisted Radiology*.
- Geldner, G., Eberhart, L. H. J., Trunk, S., Dahmen, K. G., Reissmann, T., Weiler, T., & A.Bach. 2002. Effizientes OP-Management. *Anaesthesist*, **51**, 760–767.
- Germond, J. F., & Haeffliger, J. M. 2001. Electronic dataflow management in radiotherapy: routine use of the DICOM-RT protocol. *Journal de la Société française de Cancer radiothérapie*, **5**(supp 1), 172– 180.
- Gessat, M., Zachow, S., Lemke, H. U., & Burgert, O. 2007 (June). Geometric Meshes in Medical Applications - Steps towards a Specification of Geometric Models in DICOM. In: Lemke, H. U., Inamura, K., Doi, K., Vannier, M. W., & Farman, A. G. (eds), *International Journal of Computer Assisted Radiology and Surgery*, vol. 2, supp. 1.
- Gessat, M., Merk, D. R., Falk, V., Walther, T., Jacobs, S., Nöttling, A., & Burgert, O. 2009. A Planning System for Transapical Aortic Valve Implantation. In: Miga, M. I., & Wong, K. H. (eds), *SPIE Medical Imaging: Visualization, Image-guided Procedures and Modeling*. Presented at the SPIE Conference, vol. 7261, paper 7261-24.
- Gildenberg, P. L., Lozano, A. M., Krauss, J. K., Hamani, C., Linderorth, B., Benabid, A. L., Seigneuret, E., Broggi, G., Franzini, A., & Velasco, F. 2009. *Textbook of Stereotactic and Functional Neurosurgery*. Springer. ISBN 978-3-540-69959-0.
- Gohr, H., & Wedekind, T. 1940. Der Ultraschall in der Medizin. *Klinische Wochenschrift*, **19**(2), 25–29.

- Goldman, A. P., Kotler, M. N., Scanlon, M. H., Ostrum, B., Parameswaran, R., & Parry, W. R. 1986. The complementary role of magnetic resonance imaging, next term doppler echocardiography, and computed tomography in the diagnosis of dissecting thoracic aneurysms. *American Heart Journal*, **111**(5), 970–981.
- Gong, S. J., O’Keefe, G. J., & Scott, A. M. 2005. Comparison and Evaluation of PET/CT image registration. *Pages 1599–1603 of: Proc. of the IEEE Annual Conference on Engineering in Medicine and Biology*, vol. 27.
- Gray, H. 1918. *Anatomy of the human body*. 20. edn. Philadelphia: Lea & Febiger. ISBN 1-58734-102-6.
- Gregorie, E. M., & Goldring, S. 1984. Localization of function in the excision of lesions from the sensorimotor region. *Journal of neurosurgery*, **61**(6), 1047–1054.
- H. U. Lemke (ed.). 2006. *White Paper of DICOM Working Group 24*. [Work in progress, obtain draft through Working Group 24.].
- Haller, J. W., Clarke, L., & Hamilton, B. 2002. *Report of the NIH/NSF Group on Image-Guided Interventions*. Tech. rept. National Cancer Institute, National Institute of Biomedical Imaging and Bioengineering, National Science Foundation.
- Hanna, K. J. 1983. Graphic analysis of pharmacy workload as a management tool. *Hospital Pharmacy*, **18**(6), 299–301.
- Hemm, S., Rigau, V., Chevalier, J., Picot, M. C., Bauchet, L., El Fertit, H., Rodriguez, M.-A., Cif, L., Vayssiere, N., Zanca, M., Baldet, P., Segnarbieux, F., & Coubes, P. 2005. Stereotactic Coregistration of 201Tl SPECT and MRI Applied to Brain Tumor Biopsies. *The Journal of Nuclear Medicine*, **46**(7), 1151–1157.
- Higgins, G., Athey, B., Burgess, J., Champion, H., *et al.* 2001. Final Report of the Meeting "Modelling & Simulation in Medicine: Towards an Integrated Framework". *Computer Aided Surgery*, **6**, 32–39.
- Himbert, D., Descoutures, F., Al-Attar, N., Iung, B., Ducrocq, G., Détaint, D., Brochet, E., Messika-Zeitoun, D., Francis, F., Ibrahim, H., Nataf, P., & Vahanian, A. 2009. Results of transfemoral or transapical aortic valve implantation following a uniform assessment in high-risk patients with aortic stenosis. *Journal of the American College of Cardiology*, **54**(4), 303–311.
- Hinshaw, W. S., Bottomley, P. A., & Holland, G. N. 1977. Radiographic thin-section image of the human wrist by nuclear magnetic resonance. *Nature*, **270**, 722–723.
- HL7, Inc. 1987. *Health Level Seven (HL7) Version 2.x*.
- HL7, Inc. 2005. *Health Level Seven (HL7) Version 3*.
- Horii, S. C., & Bidgood, W. D. 1992. PACS mini refresher course. *RadioGraphics*, **12**, 537–548.

- Huang, H. K. 2004. *PACS and Imaging Informatics: Basic Principles and Applications*. 2 edn. Wiley-Liss. ISBN 0471251232.
- Ibach, B., Zimolong, A., Portheine, F., Niethard, F.U., & Radermacher, K. 2006 (June). Integrated medical workstations for Computer Integrated Smart Surgery (CISS) - state of the art, bottlenecks and approaches. In: *International Journal of Computer Assisted Radiology and Surgery*, vol. 1, supp. 1.
- IEEE. 1985. *IEEE754 - IEEE Standard for binary floating-point arithmetic*. <http://754r.ucbtest.org/standards/754.pdf>. [Online; accessed 25-October-2009].
- IHE. 2009. *Integration the Healthcare Enterprise*. www.ihe.net. [Online; accessed 03-December-2009].
- ISO. 2006. *ISO 11073 - Health informatics – Point-of-care medical device communication*.
- Jabbour, P., Tjournakaris, S., & Rosenwasser, R. 2009. *Angiography, MRA in Image Guided Neurosurgery*. Springer. Chap. 21, pages 299–305.
- Jacobs, S., Merk, D. R., Holzey, D., & Falk, V. 2008. Modellbasierte Therapie in der Herzchirurgie. *Pages 132–139 of: Niederlag, W., Lemke, H. U., Meixensberger, J., & Baumann, M. (eds), Modellgestützte Therapie*. Health Academy.
- Jannin, P., & Morandi, X. 2007. Surgical models for computer-assisted neurosurgery. *NeuroImage*, **37**, 783–791.
- Jenkinson, M., Pechaud, M., & Smith, S.M. 2005. BET2: MR-based estimation of brain, skull and scalp surfaces. *Eleventh Annual Meeting of the Organization for Human Brain Mapping*.
- Jolesz, F. A., & Shtern, F. 1992. The operating room of the future. Report of the National Cancer Institute Workshop, "Imaging-Guided Stereotactic Tumor Diagnosis and Treatment". *Investigative Radiology*, **27**(2), 326–328.
- Jolesz, F. A., Nabavi, A., & Kikinis, R. 2001. Integration of Interventional MRI with Computer-Assisted Surgery. *Journal of Magnetic Resonance Imaging*, **13**, 69 – 77.
- Jolesz, F.A. 1997. Image-guided procedures and the operating room of the future. *Radiology*, **204**(3), 601–612.
- Kalman, R. E. 1960. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, **82**(Series D), 35–45.
- Kangarloo, H., Dietrich, R. B., Ehrlich, R. M., Boechat, M. I., & Feig, S. A. 1986. Magnetic resonance imaging next term of wilms tumor. *Urology*, **28**(3), 203–207.

- Karar, M. E., Chalopin, C., Merk, D. R., Jacobs, S., Walther, T., Falk, V., & Burgert, O. 2009. Localization and tracking of aortic valve prosthesis in 2D fluoroscopic image sequences. *In: Miga, M. I., & Wong, K. H. (eds), SPIE Medical Imaging: Visualization, Image-guided Procedures and Modeling*. Presented at the SPIE Conference, vol. 7261, paper 7261-1Q.
- Kaufman, A., & Wang, J. 2002. 3D Surface Reconstruction from Endoscopic Videos. *Pages 61–74 of: Linsen, L., Hagen, H., & Hamann, B. (eds), Visualization in Medicine and Life Sciences*. Mathematics and Visualization. Springer Berlin Heidelberg.
- Kawamata, T., Iseki, H., Shibasaki, T., & Hori, T. 2002. Endoscopic Augmented Reality Navigation System for Endonasal Transsphenoidal Surgery to Treat Pituitary Tumors: Technical Note. *Neurosurgery*, **50**(6), 1393–1397.
- Kazi, Z., Bakharev, V.A., & Stygar, A.M. 1979. Znachenie ul'trazvukovogo issledovaniia pri biopsii khoriona po geneticheskim pokazaniiam (Value of the ultrasonic studies in biopsy of the chorion, according to genetic indicators). *Akusherstvo i ginekologiya*, **8**, 29–31.
- Kettenbach, J., Jolesz, F. A., & Kikinis, R. 1997. Surgical Planning Laboratory: a new challenge for radiology? *Pages 855–860 of: Proceedings of the 11th International Symposium and Exhibition of Computer Assisted Radiology and Surgery*.
- Kikinis, R., Gleason, P. L., & Jolesz, F. A. 1996. Surgical Planning Using Computer-Assisted Three-Dimensional Reconstructions. *Chap. 8, pages 147–154 of: Taylor, R. H., Lavallée, S., Burdea, G. C., & Mösges, R. (eds), Computer Integrated Surgery - Technology and Clinical Applications*. The MIT Press.
- Kim, K. S., & Weinberg, P. E. 1986. Magnetic Resonance Imaging of a Spinal Extradural Arachnoid Cyst. *Surgical Neurology*, **26**, 249–252.
- Knauth, M., Wirtz, C. R., Tronnier, V. M., Aras, N., Kunze, S., & Sartor, K. 1999. Intraoperative MR Imaging Increases the Extent of Tumor Resection in Patients with High-Grade Gliomas. *AJNR Am J Neuroradiol*, **20**(9), 1642–1646.
- Knuth, D. E. 1998. *Sorting and Searching*. Second edn. The Art of Computer Programming, vol. 3. Reading, Massachusetts: Addison-Wesley.
- Kosugi, Y., Watanabe, E., Goto, J., Watanabe, T., Yoshimoto, S., Takakura, K., & Ikebe, J. 1988. An articulated neurosurgical navigation system using MRI and CT images. *IEEE Transactions on Biomedical Engineering*, **35**(2), 147–152.
- Kouchoukos, N., Blackstone, E., Doty, D., Hanley, F., & Karp, R. 2003. *Kirklin/Barratt-Boyes Cardiac Surgery*. 3rd edn. Churchill Livingstone.
- Koulechov, K., Strauß, G., Richter, R., Trantakis, C., & Lüth, T. 2005. Mechatronical assistance for paranasal sinus surgery. *In: Lemke, H. U., Inamura, K., Doi, K.,*

- Vannier, M. W., & Farman, A. G. (eds), *Computer Assisted Radiology and Surgery*, vol. 19.
- Kubota, T., Takeuchi, H., Handa, Y., & Sato, K. 2004. Application of mobile CT for neurosurgical operation and stereotactic radiotherapy. *International Congress Series*, **1259**, 397 – 407.
- Lauterbur, P.C. 1973. Image Formation by Induced Local Interactions: Examples Employing Nuclear Magnetic Resonance. *Nature*, **242**, 190–191.
- Law, M. Y. Y., & Huang, H. K. 2003. Concept of a PACS and imaging informatics-based server for radiation therapy. *Computerized Medical Imaging and Graphics*, **27**(1), 1–9.
- Law, M. Y. Y., & Liu, B. 2009. DICOM-RT and Its Utilization in Radiation Therapy. *Radiographics*, **29**(3), 655–667.
- Law, M. Y. Y., Liu, B., & Chan, L. W. 2009. Informatics in radiology: DICOM-RT-based electronic patient record information system for radiation therapy. *Radiographics*, **29**(4), 961–972.
- Le Bihan, D., Mangin, J.-F., Poupon, C., Clark, C. A., Pappata, S., Molko, N., & Chabriet, H. 2001. Diffusion tensor imaging: Concepts and applications. *Journal of Magnetic Resonance Imaging*, **13**(4), 534–546.
- Lemke, H. U. 1985. The Third Dimension. *Pages 628–634 of: Proceedings on the International Symposium on Computer Assisted Radiology*.
- Lemke, H. U. 2007. Summary of the white paper of DICOM WG24 'DICOM in Surgery'. In: Horii, S. C., & Andriole, K. P. (eds), *SPIE Medical Imaging: PACS and Imaging Informatics*. Presented at the SPIE Conference, vol. 6516, paper 651603.
- Lemke, H. U., & Berliner, L. 2007. Specification and design of a Therapy Imaging and Model Management System (TIMMS). In: Horii, S. C., & Andriole, K. P. (eds), *SPIE Medical Imaging: PACS and Imaging Informatics*. Presented at the SPIE Conference, vol. 6516, paper 651602.
- Lemke, H. U., & Berliner, L. 2008. Modellgestützte Therapie, patientenspezifisches Modell und modellbasierte Evidenz. *Pages 13–24 of: Niederlag, W., Lemke, H.U., Meixensberger, J., & Baumann, M. (eds), Modellgestützte Therapie*. Health Academy, vol. 13.
- Lemke, H. U., & Vannier, M. W. 2006. The operating room and the need for an IT infrastructure and standards. *International Journal of Computer Assisted Radiology and Surgery*, **1**(3), 117–121.

- Lemke, H. U., Faulkner, G., & Krauss, Ma. 1994. Developments Towards Multimedia Medical Workstations. *Computerized Medical Imaging and Graphics*, **18**(2), 61 – 67.
- Lemke, H. U., Trantakis, C., Köchy, K., Müller, A., Strauß, G., & Meixensberger, J. 2004. Workflow analysis for mechatronic and imaging assistance in head surgery. *International Congress Series*, **1268**, 830 – 835. CARS 2004 - Computer Assisted Radiology and Surgery. Proceedings of the 18th International Congress and Exhibition.
- Leven, J., Burschka, D., Kumar, R., Zhang, G., Blumenkranz, S., Dai, X., Awad, M., Hager, G. D., Marohn, M., Choti, M., Hasser, C., & Taylor, R. H. 2005. DaVinci Canvas: A Telerobotic Surgical System with Integrated, Robot-Assisted, Laparoscopic Ultrasound Capability. *Pages 811–818 of: Duncan, J., & Gerig, G. (eds), Proc. of Medical Image Computing and Computer-Assisted Intervention - MICCAI. Lecture Notes in Computer Science*, vol. 3749/2005.
- Lindner, D., Trantakis, C., Schmidtgen, A., Grunst, G., Arnold, S., & Meixensberger, J. 2003. Iterative neuronavigation using 3D ultrasound—a feasibility study. *Pages 619 – 624 of: Proceedings on the International Symposium on Computer Assisted Radiology and Surgery*, vol. 17.
- Löpfe, A., Stoeckle, U., & Joskowicz, L. 2006. *White Paper of DICOM Working Group 24 - Chapter 5: Workflow and Medical Imaging in Orthopaedic Surgery*. [Work in progress, obtain draft through Working Group 24.].
- Lüth, T., Bier, J., Bier, A., & Hein, A. 2001. *Verfahren und Gerätesystem zum Materialabtrag oder zur Materialbearbeitung*. Patent DE. 101 17 403 C2. 2001.
- Mack, M., Acuff, T., Yong, P., Jett, G. K., & Carter, D. 1997. Minimally invasive thoroscopically assisted coronary artery bypass surgery. *European Journal of Cardio-Thoracic Surgery*, **12**, 20–24.
- Maintz, J. B., & Viergever, M. A. 1998. A survey of medical image registration. *Medical Image Analysis*, **2**(1), 1–36.
- Manwaring, K. H., Manwaring, M. L., & Moss, S. D. 1994. Magnetic field guided endoscopic dissection through a burr hole may avoid more invasive craniotomies. A preliminary report. *Acta Neurochirurgica. Supplement*, **61**, 34–39.
- Marzullo, K. A. 1984. *Maintaining the Time in a Distributed System: An Example of a Loosely-Coupled Distributed Service*. Stanford University, Department of Electrical Engineering.
- Meixensberger, J. 2008. Modellbasierte Therapie – Einfluss und Auswirkungen auf das Betätigungsfeld des Chirurgen. *Pages 271–277 of: Niederlag, W., Lemke, H. U., Meixensberger, J., & Baumann, M. (eds), Modellgestützte Therapie*. Health Academy.

- Meyer, B. 2008. *Modularity*. Prentice Hall. Chap. B.3, pages 1–21.
- Meyer, F. B., Bates, L. M., Goerss, S. J., Friedman, J. A., Windschitl, W. L., Duffy, J. R., Perkins, W. J., & O'Neill, B. P. 2001. Awake craniotomy for aggressive resection of primary gliomas located in eloquent brain. *Mayo Clinic Proceedings*, **76**(7), 677–687.
- Meyer, M., Levine, W. C., Brzezinski, P., Robbins, J., Lai, F., Spitz, G., & Sandberg, W. S. 2003. Integration of Hospital Information Systems, Operative and Peri-operative Information Systems, and Operative Equipment into a Single Information Display. *Page 1054 of: Proc. of AMIA 2005 Symposium*.
- Morgan, Jr., G. E., Mikhail, M. S., & Murray, Michael J. 2001. *Clinical Anaesthesiology*. McGraw-Hill Professional. ISBN 0-838-51553-3.
- Mould, R. F. 1980. *A history of X-rays and radium with a chapter on radiation units: 1895-1937*. IPC Building & Contract Journals, Sutton :.
- Mozer, P., Leroy, A., Y. Payan and, J. Troccaz, Chartier-Kastler, E., & Richard, F. 2006. Computer-assisted access to the kidney. *The International Journal of Medical Robotics and Computer Assisted Surgery*, **1**(4), 58–66.
- Mösgeles, R. 1993. New trends in head and neck imaging. *European Archives of Oto-Rhino-Laryngology*, **250**, 317–326.
- Mudunuri, R., Neumuth, T., Strauß, G., Dietz, A., Meixensberger, J., & Burgert, O. 2007 (June). SOCAS - Surgical Ontologies for Computer Assisted Surgery. *In: Lemke, H. U., Inamura, K., Doi, K., Vannier, M. W., & Farman, A. G. (eds), International Journal of Computer Assisted Radiology and Surgery*, vol. 2, supp. 1.
- Muller, R.N., Marsh, M.J., Bernardo, M.L., & Lauterbur, P.C. 1982. True 3-D imaging of limbs by NMR zeugmatography with off-resonance irradiation. *European Journal of Radiology*, **3**(suppl. 1), 286–290.
- Nakamura, S., Colombo, A., Gaglione, A., Almagor, Y., Goldberg, S. L., Maiello, L., Finci, L., & Tobis, J. M. 1994. Intracoronary ultrasound observations during stent implantation. *Circulation*, **89**(5), 2026–2034.
- NEMA. 1993. *DICOM Supplement 10: Basic Worklist Management - Modality*. ftp://medical.nema.org/medical/dicom/final/sup10_ft.pdf. [Online; accessed 03-December-2009].
- NEMA. 1996. *DICOM Supplement 11: Radiotherapy Information Objects*. ftp://medical.nema.org/medical/dicom/final/sup11_ft.pdf. [Online; accessed 25-October-2009].
- NEMA. 2008a. *DICOM Supplement 132: Surface Segmentation*. ftp://medical.nema.org/medical/dicom/final/sup132_ft.pdf. [Online; accessed 30-October-2009].

- NEMA. 2008b. *Digital Imaging and Communications in Medicine (DICOM)*. <http://dicom.nema.org>. [Online; accessed 25-October-2009].
- NEMA. 2009a. *DICOM Supplement 131: Implant Templates*. <http://www.dclunie.com/dicom-status/status.html#SupplementsByNumber>. [Online (work in progress); accessed 30-October-2009].
- NEMA. 2009b. *DICOM Supplement 134: Implant Planning SR Document*. <http://www.dclunie.com/dicom-status/status.html#SupplementsByNumber>. [Online; accessed 30-October-2009].
- Neumuth, T., Pretschner, A., Trantakis, C., Fischer, M., Lemke, H. U., & Burgert, O. 2005. Workflow-analysis of Surgical Interventions in ENT and Neurosurgery. *Pages 85–86 of: Computer Aided Surgery around the Head - 3rd International Symposium Proceedings*, vol. 258.
- Neumuth, T., Jannin, P., Strauß, G., Meixensberger, J., & Burgert, O. 2009. Validation of Knowledge Acquisition for Surgical Process Models. *Journal of the American Medical Informatics Association*, **16**, 72–80.
- Nezhat, C. 2005. *Nezhat's History of Endoscopy*. <http://laparoscopy.blogs.com/endoscopyhistory/>. [Online; accessed 18-September-2009].
- Nezhat, C., Crowgey, S., & Garrison, C. 1986. Surgical treatment of endometriosis via laser laparoscopy. *Fertility and Sterility*, **45**(6), 778–783.
- Niederlag, W., Lemke, H.U., Meixensberger, J., & Baumann, M. (eds). 2008. *Modellgestützte Therapie*. Health Academy, vol. 13.
- NifTI. 2005. *Neuroimaging Informatics Technology Initiative: NifTI-1 Data Format*. <http://nifti.nimh.nih.gov/>. [Online; accessed 27-April-2009].
- NLM. 1989. *The Visible Human Project*. http://www.nlm.nih.gov/research/visible/visible_human.html. [Online; accessed 10-December-2009].
- NSR . 2000. *The Physiome Project*. <http://www.physiome.org>. [Online; accessed 10-December-2009].
- Nuttin, B., Knauth, M., Gybels, J., Verbeeck, R., Vandermeulen, D., Michiels, J., Suetens, P., & Marchal, G. 1994. How does the stereotactic workstation help the neurosurgeon? *Stereotactic and functional neurosurgery*, **63**(1-4), 71–22.
- Ohnuma, K., Masamuneb, K., Yoshimitsua, K., Sadahiroa, T., Vainc, J., Fukuia, Y., & Miyawakia, F. 2006 (June). Timed-automata-based model for laparoscopic surgery and intraoperative motion recognition of a surgeon as the interface connecting the surgical scenario and the real operating room. *In: International Journal of Computer Assisted Radiology and Surgery*, vol. 1, supp. 1.

- Okudera, H. 2000. Intraoperative angiography for emergency cerebrovascular surgery using an exclusively developed radiolucent Sugita head frame and fixation. *Journal of Clinical Neuroscience*, **7**(6), 539 – 541.
- Paieon. 2009. *C-THV*. <http://www.paieon.com>. [Online; accessed 10-December-2009].
- Patkin, M. 2003. What surgeons want in operating rooms. *Minimally Invasive Therapy and Allied Technologies*, **12**(6), 256–262.
- Peters, T. M., Clark, J., Pike, B., Drangova, M., & Olivier, A. 1987. Stereotactic Surgical Planning with Magnetic Resonance Imaging, Digital Subtraction Angiography and Computed Tomography. *Applied Neurophysiology*, **50**(1-6), 33 – 38.
- Pham, D. L., Xu, C., & Prince, J. L. 2000. A Survey of Current Methods in Medical Image Segmentation. *Annual Review of Biomedical Engineering*, **2**, 315–338.
- Pianykh, O. S. 2008. *Digital Imaging and Communications in Medicine (DICOM): A Practical Introduction and Survival Guide*. Springer. ISBN 978-3-540-74570-9.
- Pillay, P. K., Barnett, G., & Awad, I. 1992. MRI-guided stereotactic placement of depth electrodes in temporal lobe epilepsy. *British Journal of Neurosurgery*, **6**(1), 47–53.
- Prokosch, H.U., & Dudeck, J. 1995. *Hospital Information Systems: Design and Development Characteristics; Impact and Future Architecture*. Medical Artificial Intelligence, vol. 2.
- Raimbault, M., Jannin, P., Morandi, X., Riffaud, L., & Gibaud, B. 2009. Models of surgical procedures for multimodal image-guided neurosurgery. *Studies in Health Technology and Informatics*, **95**, 50–55.
- Reinhardt, H.F., Horstmann, G.A., & Gratzl, O. 1993. Sonic stereometry in microsurgical procedures for deep-seated brain tumors and vascular malformations. *Neurosurgery*, **32**(1), 51–57.
- Riedl, S. 2002. Modernes Operationsmanagement im Workflow Operation. *Chirurg*, **73**, 105–110.
- Röntgen, W. C. 1895. Über eine neue Art von Strahlen (Vorläufige Mittheilung). In: *Sitzungsberichte der Würzburger Physik.-medic. Gesellschaft 1895*. Verlag der Stachel'schen k. Hof- u. Univers.-Buch- u. Kunsthandlung.
- Roberts, D. W., Strohbehn, J. W., Hatch, J. F., Murray, W., & Kettenberger, H. 1989. A frameless stereotaxic integration of computerized tomographic imaging and the operating microscope. *Journal of Neurosurgery*, **65**(4), 545–549.
- Rogers, E. M. 2003. *Diffusion of Innovations*. 5 edn. New York, London, Toronto, Sydney: The Free Press.

- Rogowska, J. 2000. Overview and Fundamentals of Medical Image Segmentation. *Chap. 5, pages 69–86 of: Bankman, I. (ed), Computer Integrated Surgery - Technology and Clinical Applications*. Academic Press.
- Romstock, J., Fahlbusch, R., Ganslandt, O., Nimsky, C., & Strauß, C. 2002. Localisation of the sensorimotor cortex during surgery for brain tumours: feasibility and waveform patterns of somatosensory evoked potentials. *British Medical Journal*, **72**(2), 221–229.
- Rossi Mori, A., Gangemi, A., Steve, G., Consorti, F., & Galeazzi, E. 1997. An Ontological Analysis of Surgical Deeds. *Pages 361–372 of: Proceedings of the 6th Conference on Artificial Intelligence in Medicine in Europe*.
- Rygh, O. M., Nagelhus Hernes, T. A., Lindseth, F., Selbekk, T., Brostrup Müller, T., & Unsgaard, G. 2006. Intraoperative navigated 3-dimensional ultrasound angiography in tumor surgery. *Surgical Neurology*, **66**(6), 581 – 592.
- Sandberg, W. S., Ganous, T. J., & Steiner, C. 2003. Setting a Research Agenda for Perioperative Systems Design. *Surgical Innovation*, **10**(2), 57–70.
- Satava, R. M. 2003. Robotic Surgery: from past to future - a personal journey. *The Surgical Clinics of North America*, **83**(6), 1491–1500.
- Schlöndorff, G. 1998. Computer-assisted surgery: Historical remarks. *Computer Aided Surgery*, **3**(4), 150–152.
- Schrader, U., Kotter, E., Pelikan, E., Zaiß, A., Timmermann, U., & Klar, R. 1997. Critical Success Factors for a Hospital-wide PACS. *Proceedings of the AMIA Annual Fall Symposium*, 439–443.
- Schroeder, W., Martin, K., & Lorensen, B. 2004. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. 3 edn. Kitware, Inc.
- Scott-Conner, C. E. H., & Berci, G. 1993. Unsolved problems in endoscopic surgery. *Surgical Endoscopy*, **7**, 281–282.
- Sectra. 2009. *Sectra preoperative planning solution*. <http://www.sectra.com/medical/orthopaedics/>. [Online; accessed 25-October-2009].
- Seeger, W., & Zentner, J. 2002. *Neuronavigation and Neuroanatomy*. Wien-New York: Springer. ISBN 978-3-211-83741-2.
- Shahidi, R., Clarke, L., Bucholz, R. D., Fuchs, H., Kikinis, R., Robb, R. A., & Vannier, M. W. 2001. White Paper: Challenges and Opportunities in Computer-Assisted Interventions January 2001. *Computer Aided Surgery*, **6**, 176 – 181.
- Shulman, G. 1978. Semi-automatic Data Processing in Clinical Chemistry. *Clinical Biochemistry*, **11**(4), 143–147.

- Simon, R., Krieger, D., Znati, T., Lofink, R., & Sclabassi, R. J. 1995. Multimedia MedNet. *Computer*, **28**(5), 65–73.
- Skaggs, R. L. 1984. Hospital design to support greater operating efficiency. *Health Care Strategic Management*, **2**(12), 11–16.
- Spiegel, E., Wycis, H., Marks, M., & Lee, A. 1947. Stereotactic apparatus for operations on the human brain. *Science*, **106**, 349 – 350.
- Srinivasan, P., Vignes, G., Venable, C., Hazelwood, A., & Cade, T. 1984. From chart tracking to workflow management. *Proceedings of the Annual Symposium on Computer Application in Medical Care*, 884–887.
- Stachowiak, H. 1973. *Allgemeine Modelltheorie*. Springer. ISBN 3-211-81106-0.
- Staemmler, M., Brill, R., Mezer, J.U., & Gersonde, K. 1995. Principles of multimodal imaging. *Minimally Invasive Therapy and Allied Technologies*, **4**, 293 – 299.
- Stevens, J. H., Burdon, T. A., Peters, W. S., Siegel, L. C., Pompili, M. F., Vierra, M. A., St. Goar, F. G., Ribakove, G. H., Mitchell, R. S., & Reitz, B. A. 1996. Minimally invasive thoracoscopically assisted coronary artery bypass surgery. *The Journal of Thoracic and Cardiovascular Surgery*, **113**(3), 567–573.
- Stevenson, J.G., Brandestini, M.A., Weiler, T., Howard, E.A., & Eyer, M. 1979. Digital multigate Doppler with color echo and Doppler display - Diagnosis of atrial and ventricular septal defects. *Circulation*, **60**(2), 205.
- Strauß, G., Lemke, H. U., & Lüdth, T. 2008. Modellbasierte Automation in der HNO-Chirurgie - Konzeptvorstellung und Anwendungsbeispiele. *Pages 119–131 of: Niederlag, W., Lemke, H. U., Meixensberger, J., & Baumann, M. (eds), Modellgestützte Therapie*. Health Academy.
- Streitbürger, D.-P., Franke, S., Gessat, M., & Mayoral, R. 2009. Ein modulares Assistenzsystem zur intraoperativen Lokalisation des Sulcus Centralis bei Tumorsektionen nahe des Motorkortex. *In: To appear in: Informatik 2009 - Im Fokus das Leben*. Lecture Notes in Informatics.
- Taylor, R. H., Lavallée, S., Burdea, G. C., & Mösges, R.h (eds). 1996. *Introduction*. The MIT Press. Pages xiii–xix.
- Tebo, S. A., Leopold, D. A., Long, D. M., Zinreich, S. J., & Kennedy, D. W. 1996. An optical SD digitizer for frameless stereotactic surgery. *IEEE Computer Graphics and Applications*, **16**(1), 55–64.
- Teraea, S., Miyasakab, K., Fujitab, N., & Shiratob, H. 1998. A hospital-wide PACS: 7 year experience and new development. *Computer Methods and Programs in Biomedicine*, **57**(1-2), 5–12.

- Trantakis, C., Tittgemeyer, M., Schneider, J.-P., Lindner, D., Winkler, D., Strauß, G., & Meixensberger, J. 2003. Investigation of time-dependency of intracranial brain shift and it's relation to the extent of tumour removal using intraoperative MRI. *Neurological Research*, **25**(1), 9–12.
- Trautwein, K., Voruganti, A., Grunert, R., Korb, W., Jacobs, S., & Falk, V. 2009. Evaluation of surgical cartographic navigation system for endoscopic bypass grafting on heart phantoms. *Pages 297–298 of: International Journal of Computer Assisted Radiology and Surgery*, vol. 4, supp. 1.
- Treichel, T., Liebmann, P., Burgert, O., & Gessat, M. 2010. Applicability of DICOM Structured Report for the Standardized Exchange of Implantation Plans. *International Journal of Computer Assisted Radiology and Surgery*, **5**(1), 1–9.
- Trepel, M. 2004. *Neuroanatomie–Struktur und Funktion*. 3. edn. München, Jena: Elsevier Urban & Fisher. ISBN (978-)3-437-41297-3.
- Vaillant, M., Davatzikos, C., Taylor, R. H., & Bryan, R. N. 1997. Computer-assisted access to the kidney. *Pages 467–476 of: Proc. of CVRMed-MRCAS'97*.
- Vannier, M. W., Marsh, J. L., Wang, G., Christensen, G. E., & Kane, A. A. 1996. Surgical imaging systems. *Surgical Technology International*, **5**, 35–42.
- Vindlacheruvu, R. R., Casey, A. T. H., & Thomas, D. G. T. 1999. MRI-guided stereotactic brain biopsy: a review of 33 cases. *Surgical Neurology*, **13**(2), 143 – 147.
- Vitaz, T. W., Gaskill-Shipley, M., Tomsick, T., & Tew, Jr., J. M. 1999. Utility, Safety, and Accuracy of Intraoperative Angiography in the Surgical Treatment of Aneurysms and Arteriovenous Malformations. *American Journal of Neuroradiology*, **20**(8), 1457–1461.
- Vogt, F., Krüger, S., Winter, M., Niemann, H., Hohenberger, W., Greiner, G., & Schick, C.H. 2005. Erweiterte Realität und 3-D Visualisierung für minimal-invasive Operationen durch Einsatz eines optischen Trackingsystems. *Pages 217–221 of: Meinzer, H.-P., Handels, H., Horsch, A., & Tolxdorff, T. (eds), Bildverarbeitung für die Medizin 2005. Informatik aktuell*.
- Voruganti, A., Mayoral, R., Jacobs, S., Grunert, R., Moeckel, H., & Korb, W. 2007. Surgical cartographic navigation system for endoscopic bypass grafting. *Pages 1467–70 of: Proc. of Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 1.
- Vosburg, K. G., & San José Estépar, R. 2007. Natural Orifice Transluminal Endoscopic Surgery (NOTES): An Opportunity for Augmented Reality Guidance. *Pages 485–490 of: Proc. of Medicine Meets Virtual Reality*.
- Walther, T., Falk, V., Kempfert, J., Borger, M. A., Fassl, J., Chu, M. W., Schuler, G., & Mohr, F. W. 2008. Transapical minimally invasive aortic valve implantation; the initial 50patients. *European Journal of Cardiovascular Surgery*, **33**(6), 983–988.

- Watanabe, E., Watanabe, T., Manaka, S., Mayanagi, Y., & Takakura, K. 1987. Three-dimensional digitizer (neuronavigator): new equipment for computed tomography-guided stereotaxic surgery. *Surgical Neurology*, **27**(6), 543–547.
- Wiles, A. D., Thompson, D. G., & Frantza, D. D. 2006. Accuracy assessment and interpretation for optical tracking systems. *Pages 421–432 of: Galloway, R. L. (ed), SPIE Medical Imaging: Visualization, Image-Guided Procedures, and Display*. Presented at the SPIE Conference, vol. 5367.
- Winter, A., Haufe, G., Groh, P., Pirtkien, R., & Hentschel, B. 2002. *Modellprogramm SaxTeleMed, Evaluation (Ebene 3). Evaluierungsschwerpunkt 1: Einsatz der digitalen Bildbearbeitung und Bildkommunikation für die präoperative Planung*. Tech. rept. Universität Leipzig, Institut für Medizinische Informatik, Statistik und Epidemiologie.
- Wong, D. R., Boone, R. H., Thompson, C. R., Allard, M. F., Altwegg, L., Carere, R. G., Cheung, A., Ye, J., Lichtenstein, S. V., Ling, H., & G., Webb J. 2009. Mitral valve injury late after transcatheter aortic valve implantation. *Journal of Thoracic and Cardiovascular Surgery*, **137**(6), 1547–1549.
- Woolsey, C. N., Erickson, T. C., & Gilson, W. E. 1979. Localization in somatic sensory and motor areas of human cerebral cortex as determined by direct recording of evoked potentials and electrical stimulation. *Journal of neurosurgery*, **51**(4), 476.
- Yock, P. G., Linker, D. T., White, N. W., Rowe, M. H., Selmon, M. R., Robertson, G. C., Hinohara, T., & Simpson, J. B. 1989. Clinical applications of intravascular ultrasound imaging in atherectomy. *International Journal of Cardiac Imaging*, **4**, 117–125.
- Yoo, S. K., Kim, K. M., Kim, N. H., Huh, J. M., Chang, B. C., & Cho, B. K. 1997. Design of a medical image processing software for clinical-PACS. *Yonsei Medical Journal*, **38**(4), 193–201.
- Zamorano, L. J., Nolte, L., M., Kadi A., & Z., Jiang. 1994. Interactive Intraoperative Localization Using an Infrared-Based System. *Stereotact Funct Neurosurg*, **63**, 84–88.
- Zhang, H., Banovac, F., Lin, R., Glossop, N., Wood, B. J., Lindisch, D., Levy, E., & Cleary, K. 2006. Electromagnetic tracking for abdominal interventions in computer aided surgery. *Computer Aided Surgery*, **11**(3), 127–136.
- Zitova, B., & Flusser, J. 2003. Image registration methods: a survey. *Image and Vision Computing*, **21**, 977–1000.